НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ *Сергій СТІРЕНКО*
(підпис)
“___”_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп’ютерних систем»

спеціальності 121 «Інженерія програмного забезпечення»

на тему: «*Система автоматизованого формування й аналізу повідомлень*».

Виконав:
студент _IV_курсу, групи__ІП-64__
Вовк Антон Ігорович _____
(підпис)

Керівник:
Асистент
Сергієнко Анастасія Анатоліївна _____
(підпис)

Консультант з нормоконтролю:
Професор, доктор технічних наук
Сімоненко Валерій Павлович _____
(підпис)

Рецензент

_____
(підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМ. ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Сергій СТІРЕНКО*
(підпис)
"___"_____20__ р.

ЗАВДАННЯ
на дипломний проєкт студенту

Вовк Антон Ігорович (прізвище, ім'я, по батькові)

1. Тема проєкту «*Система автоматизованого формування й аналізу повідомлень»*
керівник проекту (роботи) асист. Сергієнко А.А.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «07» травня 2020 року №1081-С
2. Термін здачі студентом закінченого проекту (роботи) *25 червня 2020*р
3. Вихідні дані до проекту (роботи) технічна документація. теоретичні та статистичні дані з обліку даних фізичної підготовки студентів
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Опис та аналіз предметної області,аналіз наявних аналогів, дослідження розробки систем управління, конструювання архітектури та розробка системи обліку даних фізичної підготовки студентів, тестування та впровадження системи

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень)

діаграма класів системи, діаграма бази даних, блок-схема алгоритму роботи програми.

6. Консультанта проекту (робота), з вказівкою розділів роботи, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В.П.		

7. Дата видачі завдання 15.12.2019 року

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проекту (роботи)	Строк виконання етапів проекту(роботи)	Примітки
1.	Затвердження теми роботи	10.12.2019-15.12.2019	
2.	Вивчення та аналіз завдання	15.12.2020-15.03.2020	
3.	Розробка архітектури та загальної структури системи	15.03.2020-25.03.2020	
4.	Розробка структур окремих підсистем	25.03.2020-05.04.2020	
5.	Програмна реалізація системи	5.03.2020-15.04.2020	
6.	Оформлення пояснювальної записки	15.04.2020-20.05.2020	
8.	Передзахист	26.05.2020	
9.	Захист	25.06.2020	

Студент Вовк Антон Ігорович _____
(підпис)

Керівник Сергієнко Анастасія Анатоліївна _____
(підпис)

Анотація

В цьому бакалаврському дипломному проекті реалізовано систему автоматизованого формування та аналізу повідомлень.

Ця система дає змогу користувачу додавати, редагувати та видаляти адресантів, шаблони для надсилання повідомлень та ключові слова за якими буде проводитись аналіз. Користь від даної системи буде оцінена користувачами в яких є брак часу та вирішить чималу проблем у психології людини.

В цьому бакаларському дипломному проекті наявна серверна частина проекту яка роблена на мові Java за допомогою інтегрованого середовища розробки IntelliJ IDEA.

Аннотация

В этом бакалаврском дипломном проекте реализована система автоматизированного формирования и анализа сообщений.

Эта система позволяет пользователю добавлять, редактировать и удалять адресантов, шаблоны для отправки сообщений и ключевые слова по которым будет проводиться анализ. Польза от данной системы будет оценена пользователями у которых есть нехватка времени и решит большую проблем в психологии человека.

В этом бакаларському дипломном проекте имеется серверная часть проекта которая Разработана на языке Java с помощью интегрированной среды разработки IntelliJ IDEA.

Annotation

In this bachelor's degree project, a automated system for generating and analyzing messages is implemented.

This system allows the user to add, edit and delete recipients, templates for

sending messages and keywords for analysis. The benefits of this system will be appreciated by users who have a lack of time and will solve major problems in human psychology.

In this bachelor's degree project there is a server part of the project which is developed in Java using the IntelliJ IDEA integrated development environment.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

№ з/п	Формат	Позначення	Найменування	Кількість аркушів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	ІАЛЦ.467200.001 ВП	Відомість проекту	1	
3	А4	ІАЛЦ.467200.002 ТЗ	Технічне завдання	4	
4	А4	ІАЛЦ.467100.003 ПЗ	Пояснювальна записка	60	
5	А4	ІАЛЦ.467100.004 Д1	Блок-схема алгоритму	1	
6	А3	ІАЛЦ.467100.005 Д2	Діаграма бази даних	1	
7	А3	ІАЛЦ.467100.006 Д3	Діаграма класів	1	

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломної роботи

освітньо-кваліфікаційного рівня бакалавр

на тему: «Система автоматизованого формування й аналізу повідомлень»

Київ – 2020 року

ЗМІСТ

1. Найменування та область застосування

Найменування: «Система автоматизованого формування й аналізу повідомлень».

Область застосування: система для автоматизації формування та аналізу повідомлень користувачів та адресантів.

2. Підстави для розробки

Підставою для розробки системи автоматизованого формування й аналізу повідомлень є завдання на дипломне проектування, затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (НТУУ «КПІ ім. І. Сікорського»)

3. Мета та призначення розробки

Метою даної розробки є побудова системи автоматизованого формування й аналізу повідомлень. Призначення даної розробки полягає у створенні програмного забезпечення, яке дозволяє аналізувати отримані привітання від адресантів та на основі висновків формувати привітання у відповідь, у певні дати.

4. Джерела розробки

Джерелом розробки є науково-технічна література з технічних питань та статі в інтернеті за даною темою.

5. Технічні вимоги

5.1. Вимоги до розроблюваного продукту

Вимоги до розробки:

- наявність системи контролю доступу;
- редагування профілю користувача;

- редагування інформації про адресантів;

-редагування шаблонів повідомлень;

-збереження та аналіз повідомлень від адресантів;

-автоматизована відправка повідомлень через поштовий сервіс.

5.2. Вимоги до програмного забезпечення

Вимоги до програмного забезпечення:

- OS: Linux, MacOS, MS Windows;
- веб-браузери: Google Chrome, Opera, Firefox, MS Edge.

5.3. Вимоги до апаратного забезпечення

Вимоги до апаратного забезпечення:

- 32-розрядний (x86) або 64-розрядний (x64) процесор із тактовою частотою 1 ГГц або швидший;

1024 мегабайт (МБ) RAM;

6. Етапи розробки

Етапи	Термін
Вивчення літератури	16.12.2019
Складання та узгодження технічного завдання	22.12.2019
Огляд та аналіз ресурсів для розробки	15.01.2020
Проектування бази даних	23.01.2020
Розробка програмного продукту	17.02.2020
Тестування розробленої частини	07.05.2020
Налагодження та виправлення помилок	12.05.2020
Оформлення документації дипломного проекту	06.06.2020

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту

на тему: «Система автоматизованого формування й аналізу повідомлень**»**

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

MVC – Model View Controller

Tomcat – контейнер який керує життєвим циклом програмного забезпечення

Apache Jersey – контейнер який керує життєвим циклом програмного забезпечення (аналог Tomcat)

JDBC – Java Database Connectivity

JAR – Java Archive

ORM – Object-relational Mapping

XML – Extensible Markup Language

IOC – Inversion of Control

DI – Dependency injection

Principal – користувач, який успішно пройшов автентифікацію

Primary key – первинний ключ відношень в базі даних

Foreign key – зовнішній ключ відношень в базі даних

Бін – Java-клас, який знаходиться під управлінням Spring-контейнера

Persistent context – область кеш-пам'яті, яка зберігає набір сутностей, у яких зберігаються дані

ВСТУП

Кожного дня люди отримують та надсилають тисячі повідомлень. Сучасне суспільство вже не може повноцінно функціонувати без цього. Це дало людям можливість мати зв’язок з будь-якого куточка планети, у будь який час та з блискавичною швидкістю обмінюватися інформацією.

Дуже важливим є зв’язок між близькими людьми, які тимчасово за якихось обставин не можуть зустрітися. Особливо у якісь важливі дати. Сьогодні люди є дуже зайнятими, робота їх поглинає і вони забувають, який сьогодні день.

Саме тому виникає потреба автоматизації надсилання привітань. І не тільки близьким людям, а і колегам по роботі, бізнес-партнерам та просто товаришам. Чималою проблемою є образа людини через забуту дату. І це займає не останнє місце в дослідженнях з психології.

Метою даної роботи є створення автоматизованої системи формування та аналізу повідомлень, яка дасть змогу уникнути цих негативних емоцій та дасть змогу аналізувати, чи привітала людина з важливою датою, що в свою чергу надасть користувачам вибір використовувати функціонал системи для привітань тільки у випадку наявності зустрічних привітань від людини.

В результаті даного проекту була створена система, яка в майбутньому може використовуватись по всьому світу, вирішуючи велику психологічну проблему, просто за наявності доступу до мережі Інтернет.

РОЗДІЛ 1.ОГЛЯД НАЯВНИХ ON-LINE СЕРВІСІВ ДЛЯ ФОРМУВАННЯ ТА АВТОМАТИЗОВАНОЇ ВІДПРАВКИ ПРИВІТАНЬ

У цьому розділі будуть розглянуті різні сервіси для формування та автоматизованої відправки привітань. Варто зазначити, що більшість сервісів надають великий перелік послуг, які задовольняють більшість вимог з формування та візуального оформлення привітань. Також чимало сервісів надають безліч послуг з автоматизованого надсилання повідомлень у вибрані користувачем дати. Щоб перевірити ці послуги, було створено обліковий запис та вітальну листівку, яка потім була придбана та надіслана. Проходячи цей процес, малося на увазі кілька пунктів. Чи простий додаток у використанні. Як швидко користувач може створити карту, переглянути та отримати її. Як здійснюється підбір карт. Проте найголовнішим було саме, що може налаштовувати користувач. А саме автоматизовану відправку.

1.1. Порівняння наявних рішень

1.1.1. Felt

Felt- це мобільний додаток для вітальних листівок із великим вибором стилів та варіантів карт. Стилі попередньо виготовлених карток дуже варіюються. Користувач також може створити власну карту, використовуючи фотографії та застосувавши фільтри, додавши будь-яке повідомлення, яке йому подобається.

Послуги які надає сервіс: служба підписів Felt надає можливість користувачу писати картки за допомогою стилуса або пальця на мобільному пристрої. Таким чином, одержувачі побачать справжній почерк відправника, а не просто почерк сгенерований машиною. Додаток також має гарну систему

оцінювання сервісу та надає можливість користувачам залишати відгуки до будь якої послуги.

Переваги:

- велика база різноманітних карток та стилів;
- можливість створювати повідомлення, використовуючи рукопис на екрані телефона;
- персональний кабінет з усіма необхідними налаштуваннями для формування карток;
- сервіс доступний у великій кількості регіонів;
- можливість оцінювати якість послуг та залишати відгуки.

Недоліки:

- відсутність можливості автоматизованого надсилання повідомлень;
- відсутність веб-версії.

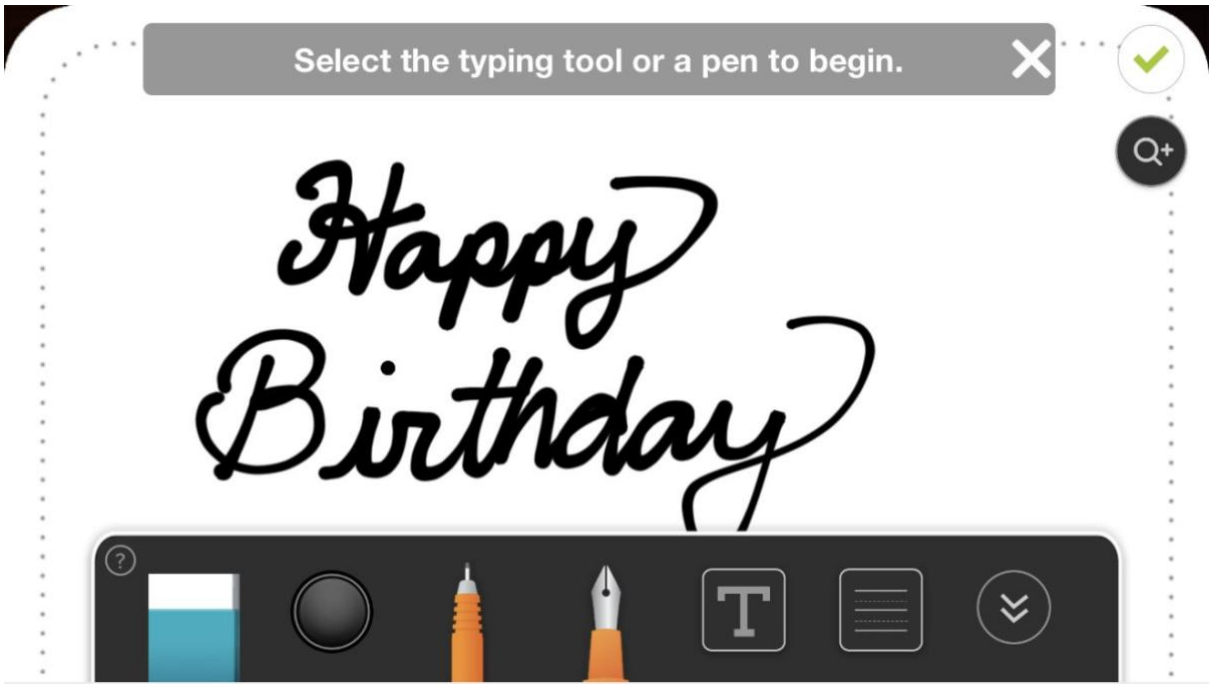


Рисунок 1.1 – Приклад формування повідомлення за допомогою власного рукопису.

1.1.2. Handwrytten

Handwrytten дає змогу створювати власні вітальні листівки зі свого мобільного додатку чи веб-сайту. Стиль карток спрямований на те, щоб бути прямолінійним, а не люб’язним чи веселим. Handwrytten позначив досить багато своїх карток як придатних для бізнесу та нерухомості, що робить його чудовим варіантом для відправки карток клієнтам. Сайт може завантажуватися повільно, але дисплей та інтерфейс працюють добре і допоможуть користувачу знайти потрібні карти.

Послуги які надає сервіс: Handwrytten спеціалізується на рукописному введенні повідомленнь на замовлених користувачем картках. Під час налаштування карток користувач може обрати потрібний стиль введення тексту, а також дату, коли користувач хоче, щоб картка надійшла на пошту.

Handwrytten інтегрується із Zapier (сервіс, який дозволяє з'єднувати тисячі веб-додатків), а це означає, що користувач може підключити його до інших додатків, якими він користуєтеся для автоматичного надсилання карти під час певної дії, наприклад, коли клієнт робить велику покупку у магазині Shopify користувача.

Основною ж особливістю даного сервісу є те, що створені користувачем листівки будуть надіслані справжнім листом у поштову скриньку.

Переваги:

- велика доступна база різноманітних карток та стилів;
- стриманий стиль оформлення привітань, що є гарним вибором для бізнесу;
- доступна веб-версія та додаток для Android та IOS;
- можливість надсилання справжніх листів у поштову скриньку;
- автоматизована відправка листівок.

Недоліки:

- сервіс доступний в дуже обмеженій кількості регіонів;
- ціна однієї вітальної картки дуже висока.

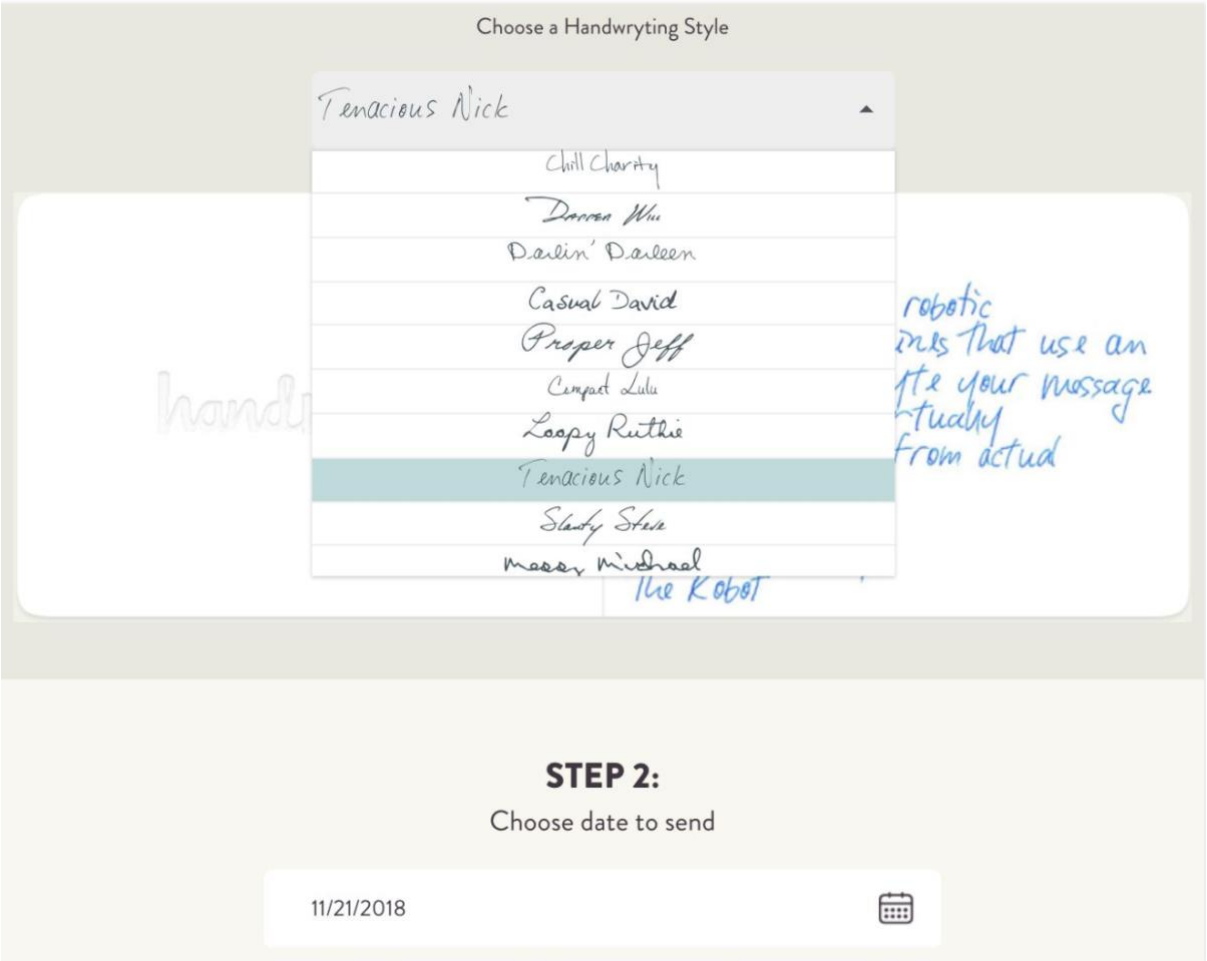


Рисунок 1.2 - Приклад вибору рукописного шрифту для привітальної картки.

1.1.3. Paperless Post Invitations

Paperless Post Invitations фокусує ціль своєї програми на зменшенні свого екологічного сліду за допомогою онлайн-запрошень. Проте залишає можливість замовляти онлайн та надсилати паперові запрошення.

Послуги, які надає сервіс: керування своїми RSVP-адресами в додатку. Оберати тисячі варіантів дизайну та ділитися будь-де за посиланням: текст, соціальні медіа чи WhatsAp. Відстежування доставки та надсилання

нагадувань про події та повідомлення гостей також доступні. Наявна автоматизована система надсилання повідомлень та оновлення списку бажаних адресантів. Окрім того, користувач може відслідковувати та відповідати на привітання.

Переваги:

- велика база різноманітних карток та стилів;
- можливість надсилати як паперову, так і цифрову версію привітання
- автоматизована відправка листівок;
- можливість відслідковувати та відповідати на привітання;
- безкоштовних привітальних карток у електронному форматі;
- моживість ділитися привітаннями у різних мессенджерах.

Недоліки:

- сервіс доступний в дуже обмеженій кількості регіонів;
- відсутність веб-версії.

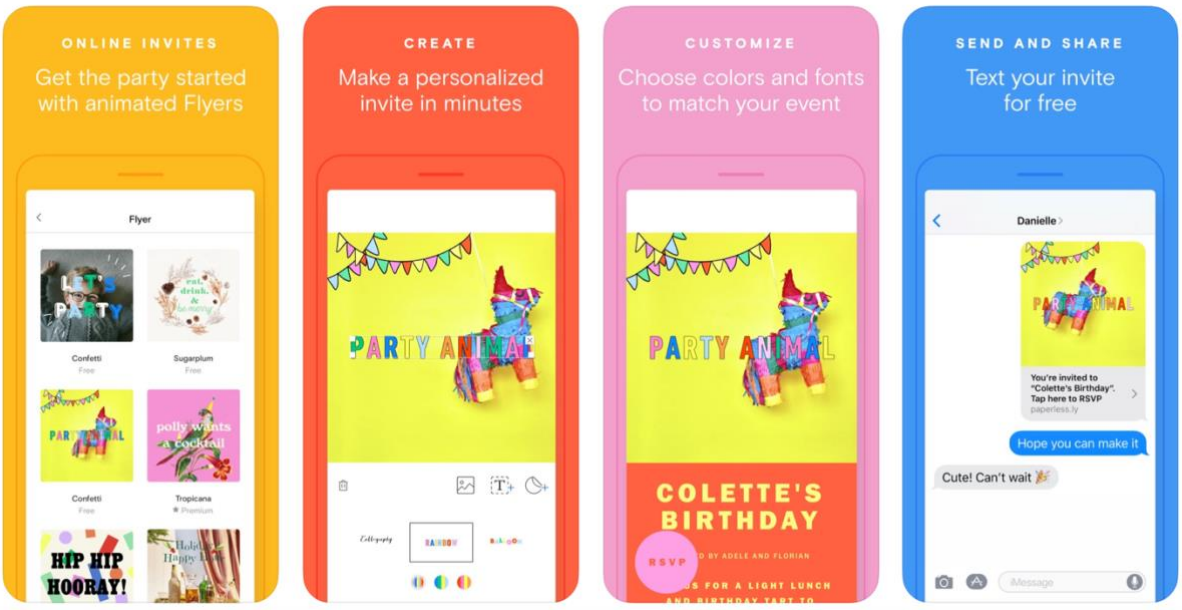


Рисунок 1.3 – Зображення підрозділів для створення привітання.

1.1.4. Ink

Ink – додаток для мобільних пристроїв, який дає змогу швидко купувати та налаштовувати вітальні листівки. У додатку є пласкі картки розміром 5 на 7 дюймів, більшість з яких можна персоналізувати зображеннями і текстом. У кількох картках є конструкції, які не потребують зображення, але більшість - так. Наявна інформаційна піктограма, яка підказує користувачу, чи обрана карта матова або глянсована.

Вибір карток спрямований на ділових людей та бізнес, тому не підходить користувачам які шукають картки із зображеннями. Багато карток у цьому каталозі мають кольорові параметри, що дає змогу обрати, скажімо, колір тла з кількох варіантів.

Послуги які надає сервіс: хоча Ink має веб-сайт із цільовою сторінкою, все одно доводиться використовувати його мобільний додаток для налаштування та придбання карток. Інтерфейс працює добре, але орієнтуватися, якщо у користувача є проблеми з мобільністю, буде важко.

Переваги:

- велика база різноманітних карток та стилів;
- можливість надсилати як паперову, так і цифрову версію привітання;
- э мобільна та веб-версія додатка;
- надсилання карток у друкованому форматі за дуже стислі часові рамки.

Недоліки:

- сервіс доступний у дуже обмеженій кількості регіонів;
- відсутність повноцінної веб-версії, що змушує для повноцінного налаштування та формування листівок завантажувати мобільний додаток;
- орієнтованість тільки на ділових осіб та бізнес;
- відсутність автоматизованого надсилання листівок.

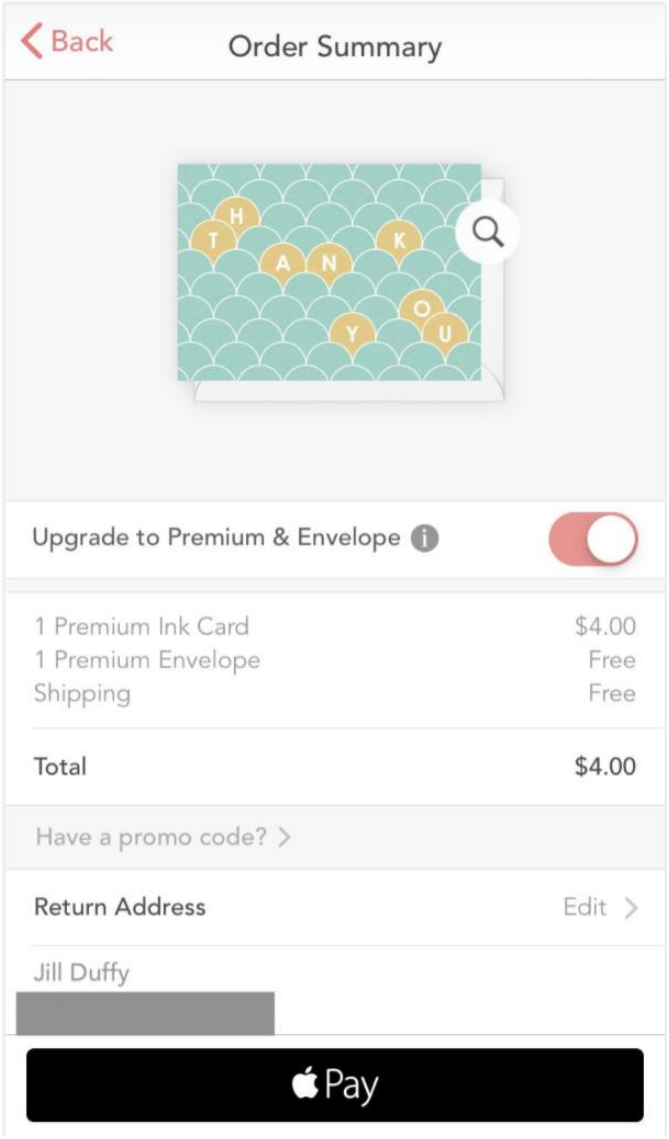


Рисунок 1.4 - Приклад завершення замовлення з запропонованими варіантами оплати.

1.1.5. Thankster

Thankster пропонує вітальні листівки на різні випадки життя та в різних стилях. Вводячи власне повідомлення, користувач може налаштувати безліч деталей тексту, наприклад, проміжку літер та висоту рядка. Якщо користувач поспішає в пошуку правильних слів, він можете вибрати повідомлення з банку пропозицій Thankster.

З усіх тестів, які було перевірино, і програм для вітальних листівок, у «Thankster» є найстаріший інтерфейс, але він компенсує це великим масивом карток.

Послуги які надає сервіс: користувач може надіслати зразок свого оригінального почерку, щоб компанія могла створити стиль, схожий саме на стиль користувача. Thankster інтегрується із Zapier, а це означає, що користувач може підключити його до інших додатків, якими він користується, та надсилати картки, коли відбуваються певні дії. Наприклад, користувач може налаштувати його так, щоб листівки подяки надсилалися щоразу, коли користувачі закривали продаж у Salesforce (хмарний сервіс який надає можливість бізнесу краще вести переговори з клієнтами, партнерами та потенційними клієнтами).

Переваги:

- великий масив карток для вибору;
- можливість надсилати як паперову, так і цифрову версію привітання;
- дуже велика кількість налаштувань та можливість роботи з маленькими деталями.

Недоліки:

- сервіс доступний в дуже обмеженій кількості регіонів;
- відсутніть мобільної версії;
- застарілий інтерфейс;
- відсутність автоматизованого надсилання листівок.

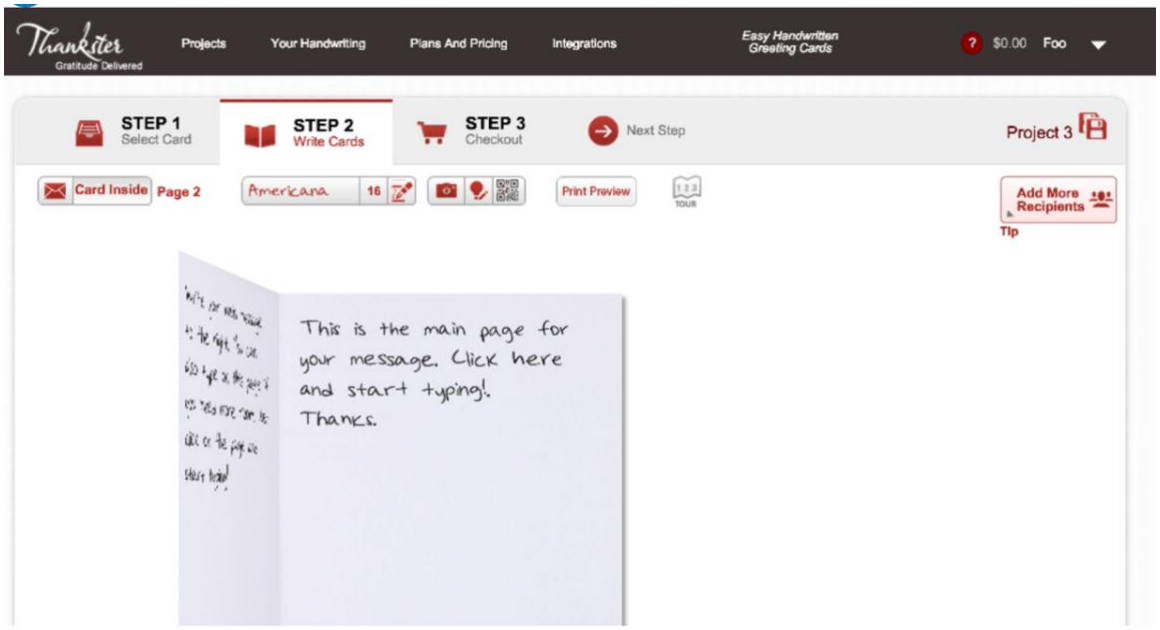


Рисунок 1.5- Приклад формування картки з великою кількістю налаштувань

ВИСНОВОК ДО РОЗДІЛУ 1

Після проведення аналізу подібніих програм було отримано висновок, що автоматизована система повинна мати наступні функції:

- Незалежність від регіону;
- Орієнтація на широку аудиторію;
- Автоматизована відправка листівок;
- Реєстрація профілю користувача;
- Можливість додавання даних про осіб, які мають отримати повідомлення;
- Система аналізу повідомлень від осіб, яких додав користувач.

РОЗДІЛ 2.ТЕХНОЛОГІЧНІ РІШЕННЯ ТА АРХІТЕКТУРА СИСТЕМИ

2.1 Вибір мови та фреймворків.

Для реалізації даного проекту була вибрана об'єктно-орієнтована мова програмування Java. Архітектурна платформа побудована за допомогою Spring Framework. Фреймворком для автоматичної збірки, компіляції та тестування використано Maven. Зберігання даних відбувається в реляційній базі даних, а провайдером обрана MySQL. Сервер здійснює зв’язок з базою даних за допомогою Spring Data. Для обміну інформацією з сервером використовується архітектура REST.

2.1.1 Spring Boot

Spring Boot - це проект, який надає встановлений набір фреймворків, щоб зменшити boilerplate configuration і запустити Spring-додаток з мінімальною кількістю коду. Тим самим позбавляючи розробника від тривалого налаштування та додавання потрібних залежностей. Та в результаті покращує процес тестування та інтеграції.

Spring Boot starters

Стартери - це велика частина Spring Boot, яка використовується для обмеження кількості налаштувань залежностей вручну, які потрібні для функціонування проекту. Стартер - це набір залежностей (в даному випадку Maven POM), специфічних для типу програми, яку представляє стартер [9].
Ось кілька популярних стартерів Spring Boot:

- spring-boot-starter-web використовується для створення RESTful веб-сервісів, використовуючи Spring MVC та Tomcat як вбудований контейнер сервлетів;
- spring-boot-starter-jerseyspring-boot-starter-jersey- це альтернатива spring-boot-starter-web, яка використовує Apache Jersey, а не Spring MVC;
- spring-boot-starter-jdbcspring-boot-starter-jdbc використовується для об'єднання з'єднань JDBC. Він заснований на впровадженні витягу з'єднань JDBC від Tomcat.

Автоконфігурація

Одна з найбільших переваг Spring Boot – це автоконфігурація. Вона починається з додавання анотації @EnableAutoConfiguration. Авто-конфігурація базується на JAR-файлах у classpath розробника та на класах відмічиних анотаціями з яких мають бути створені біни.

Переваги Spring Boot:

- розробити додатки на основі Spring за допомогою Java або Groovy набагато простіше;
- скорочує багато часу на розробку та підвищує продуктивність;
- дає змогу уникнути написання великої кількості коду, анотацій та XML-конфігурацій;
-інтегрувати додаток Spring Boot дуже просто з його екосистемою Spring, наприклад, Spring JDBC, Spring ORM, Spring Data, Spring Security тощо;
- дотримується підходу “ Opinionated Defaults Configuration ” для зменшення зусиль розробника [9];
- надає вбудовані HTTP-сервери, такі як Tomcat, Jetty тощо, щоб розробляти та тестувати наші веб-програми було легше;
- надає інструмент CLI (Command Line Interface) для розробки облегчення та тестування Spring Boot (Java або Groovy) додатків з командного рядка;

- надає безліч плагінів для розробки та тестування Spring-програм за допомогою інструментів збірок, таких як Maven і Gradle;

- надає безліч плагінів для роботи з вбудованими та in-memory базами даних.

Рисунок 2.1 – Схема роботи та конфігурації Spring фреймворка.

Основна мета Spring Boot

Основна мета Spring Boot Framework - скоротити час на розробку, тестування, інтеграцію та полегшити розробку веб-додатків, готових для розгортання на виробничій платформі порівняно з налаштовуванням Spring Framework вручну, що дійсно потребує більше часу.

Які можливості для цього наявні:

-дає змогу повністю уникнути XML конфігурації;

-дає змогу не писати багато імпортів;

-забезпечує налаштування та компоненти замовчуванням для швидкого запуску нових проектів протягом короткого часу.

Тобто Spring Boot Framework скорочує час розробки, зусилля розробника та підвищує продуктивність.

2.1.2 З чого складається Spring Framework

Головні особливості Spring Framework

IoC контейнер

Посилається на основний контейнер, використовуючи шаблон DI або IoC для надання посилання на об'єкт у класі під час виконання. Ця закономірність виступає альтернативою патерну service locator. Контейнер IoC

містить код, написаний на асемблері, який обробляє управління конфігурацією об'єктів програми.

У Spring є два пакети, а саме, org.springframework.context та org.springframework.beans, що допомагає забезпечити функціональність контейнера IoC.

Data access фреймворк

Дає змогу розробникам використовувати persistence API's, такі як Hibernate та JDBC, для зберігання даних у базі даних. Це допомагає у вирішенні багатьох проблем розробника, таких як взаємодія з підключенням до бази даних, як переконатися, що з'єднання закрилося, як боротися з виключеннями та як реалізовувати управління транзакціями. Все це дає змогу розробникам дуже легко писати код для доступу до даних.

Spring MVC фреймворк

Дає змогу створювати веб-додатки на основі MVC-архітектури. Усі запити, зроблені користувачем, спочатку проходять через контролер, а потім пересилаються на різні вигляди (View), тобто на різні сервлети або JSP сторінки. Особливості обробки та валідації форм у рамках Spring MVC можна легко інтегрувати з усіма популярними технологіями для реалізації вгляду (View), такими як FreeMarker, Velocity, ISP та Jasper Report.

Transaction managment

Допомагає в управлінні транзакціями програми, жодним чином не впливаючи на її код. Цей фреймворк пропонує API транзакцій Java (JTA) для глобальних транзакцій, під управлінням сервера, та локальних транзакцій, якими керується за допомогою JDBC Hibernate, об'єктів даних Java (JDO) або інших API для доступу до даних [5]. Це дає змогу розробнику моделювати

дуже широкий спектр транзакцій на основі декларативного та програмного управління транзакціями Spring.

Веб-сервіс Spring

Генерує кінцеві точки та визначення веб-служб на основі Java класів, але керувати ними в додатку важко. Щоб вирішити це, Веб-сервіс Spring пропонує багатошарові підходи, які окремо керується за допомогою Extensible Markup Language (XML). Spring забезпечує ефективне відображення для передачі вхідного запиту XML-повідомлення на об'єкт і завдяки цьому розробник легко зможе розподілити XML-повідомлення (об’єкта) між двома машинами.

Шар абстракції JDBC:

Допомагає розробникам легко та ефективно керувати помилками. Є можливість суттєво зменшити кількість JDBC коду, коли цей рівень абстракції реалізований у веб-додатку [6]. Цей архітектурний шар програми обробляє виключення, такі як DriverNotFound. Усі SQLExceptions переводяться в клас DataAccessException.

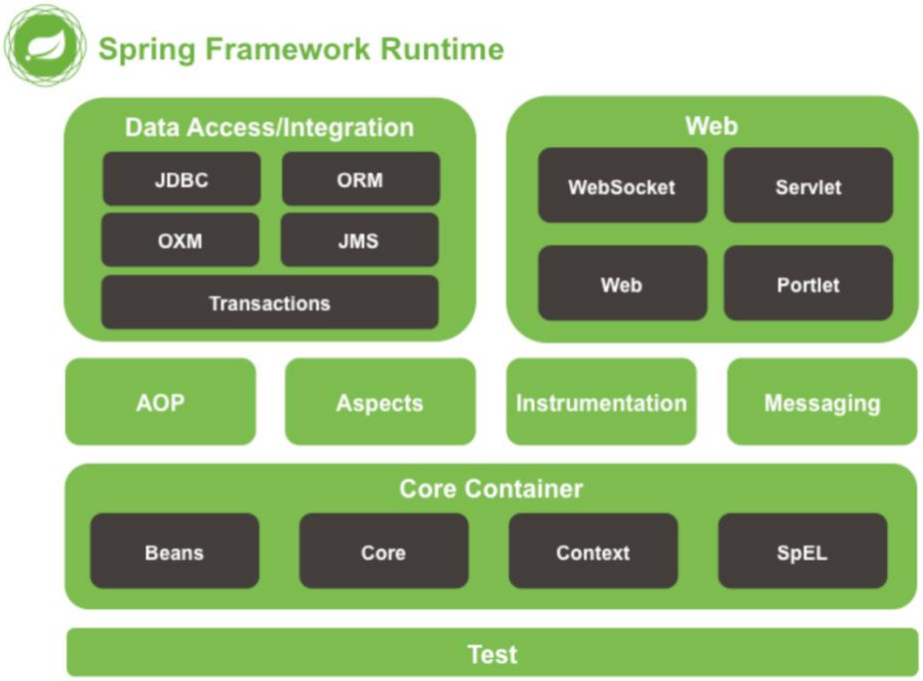


Рисунок 2.2 – компоненти Spring Framework

Переваги Spring Framework

1. Спрощення розробки Enterprise додатків. Spring дає змогу спростити розробку багатокомпонентних серверних додатків, та надає розробнику Spring Core, Spring IoC та Spring AOP для інтеграції різних компонентів бізнес-додатків.

2. Має підтримку для розробки корпоративних додатків через POJO класи.

Spring підтримує розробку Enterprise-додатків за допомогою POJO-класів, що позбавляє від потреби в імпорті важких Enterprise контейнерів під час розробки. Що значно спрощує тестування додатків.

3. Інтеграція інших фреймворків

Spring, може використовуватись з усіма іншими Java-фреймворками, розробник може використовувати ORM, Hibernate, Struts, та інші фреймворки разом. Тобто Spring не накладає жодних обмежень на кількість та комбінацію інших фреймворків які розробник може використовувати.

4. Тестування додатків

Spring container можна використовувати для проходження тестів поза Enterprise контейнером, що робить тестування значно простішим.

5. Модульність

Spring framework -це каркас який зібраний з модулів, таких як Spring MVC, Spring JDBC, Spring ORM тощо.[5]

6. Spring Transaction Management

Веб-інтерфейс, який керує транзакціями, дуже гнучкий, він може налаштовуватися на використання локальних транзакцій у невеликому додатку, який можна масштабувати до JTA для глобальних транзакцій.

2.1.3 Spring MVC

Spring MVC базується на дизайні MVC. Ось перелік ключових класів Spring MVC.

DispatcherServlet

DispatcherServlet налаштовується за допомогою файла web.xml, а необхідні шаблони URL-адрес відображаються на цьому сервлеті [10]. Він працює як контролер (Front Contoller) і обробляє всі запити, які поступають від користувача.

ModelAndView

Цей клас виступає контейнером як для моделі (Model), так і для вигляду (View) у Spring MVC.

SimpleFormController

SimpleFormController - це реалізація Concrete FormController. Він забезпечує форму, яку можна налаштовувати, та керує успішно завантаженими виглядами (View) [6].

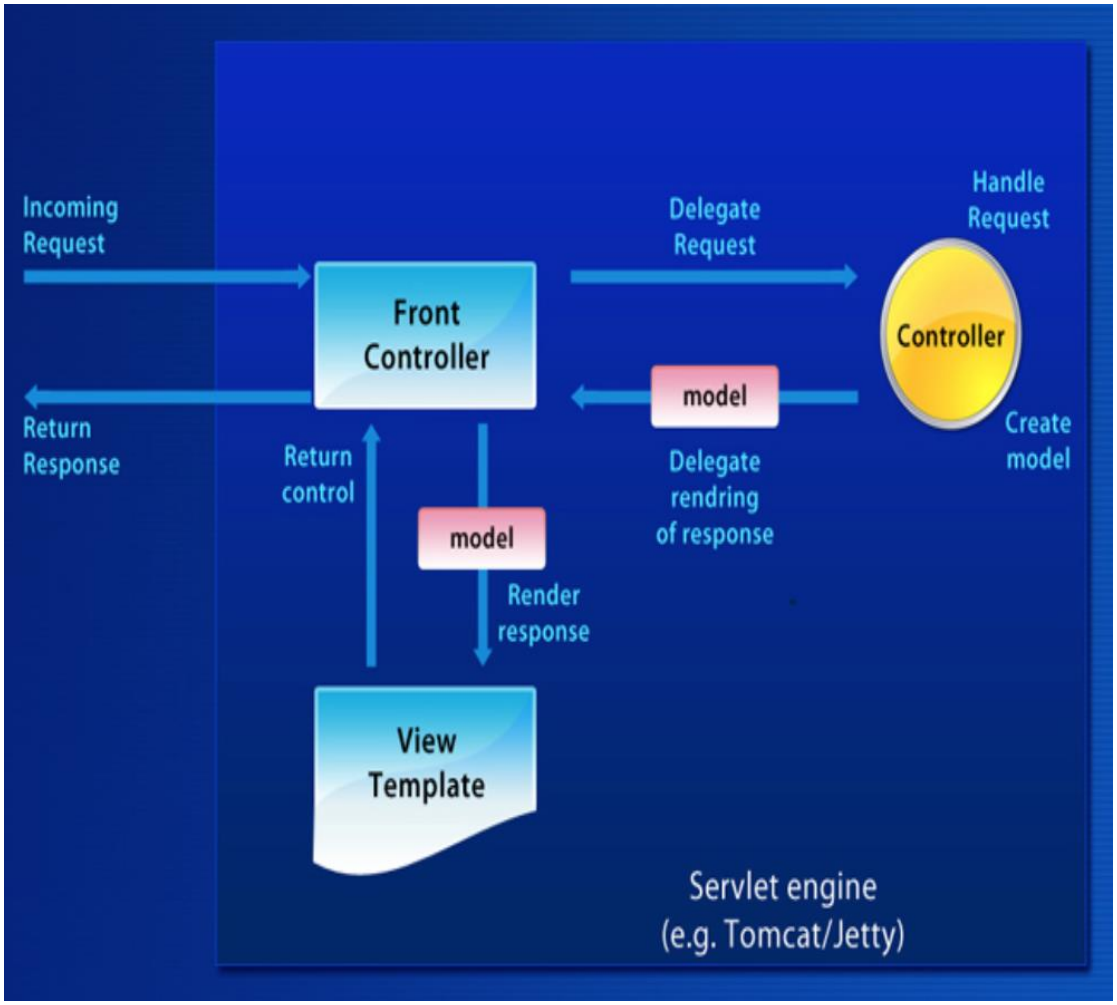


Рисунок 2.3 – спрощена схема архітектури Spring MVC.

Схема роботи MVC

1. DispatcherServlet отримує запит та звертається до інтерфейса HandlerMapping, котрий визначає, який Контролер повинен бути викликаний, після чого надсилає запит до потрібного Контролера.

2. Контролер приймає запит і викликає відповідний метод, який залежить від типу запиту [10]. Викликаний метод визначає дані Моделі, які основані на певній бізнес-логіці і повертає DispatcherServlet ім'я (View).

3. За допомогою ViewResolver DispatcherServlet визначає, котрий вигляд (View) потрібно використати на основі отриманого імені.

4. Після того, як вигляд (View) створений, DispatcherServlet надсилає дані до Моделі у вигляді артибутів у подання (View), який в кінці відображається в браузері у користувача.

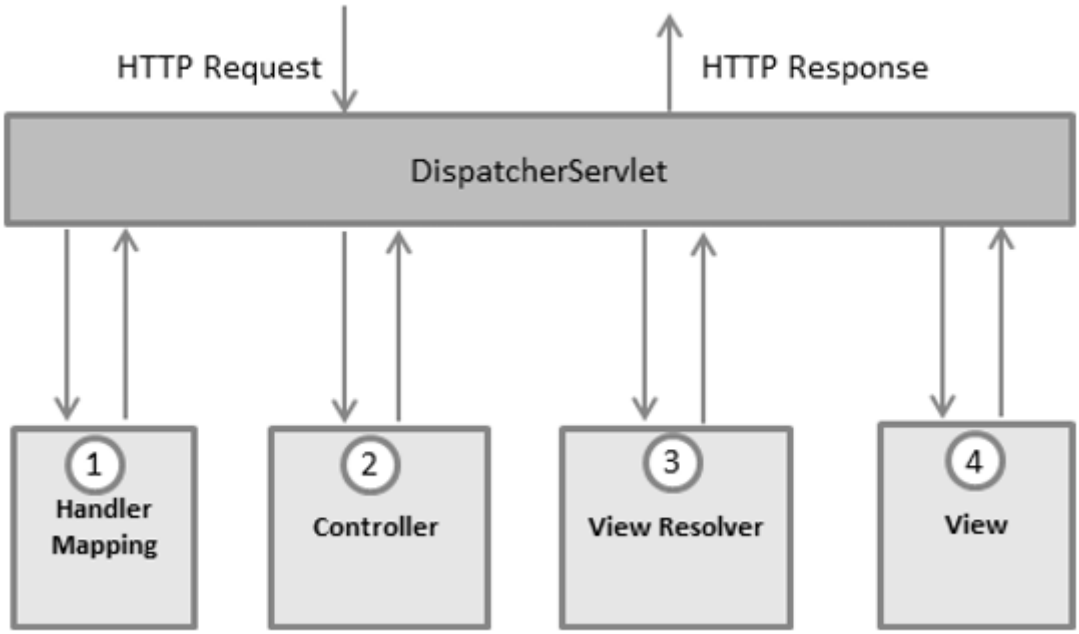


Рисунок 2.4 – Схема роботи MVC.

2.1.4 Spring security

Spring Security – це окремий модуль Spring фреймворк, який сфокусований на представленні методів для автентифікації та авторизації Java-

додатків. Він також забезпечує захист від вразливостей які частіше за все зустрічаються в безпеці, такі як CSRF attacks.

Щоб почати використовувати Spring Security в веб-додатку, треба просто вказати анотацію.: *@EnableWebSecurity*.

Автентифікація

Основним інтерфейсом автентифікації є AuthenticationManager, який має лише один метод:

```
public interface AuthenticationManager {

  Authentication authenticate(Authentication authentication)
    throws AuthenticationException;

}
```

Рис 2.5 – Інтерфейс AuthenticationManager

Метод AuthenticationManager може виконати одну з трьох речей у своєму методі автентифікації.

1.Повернути автентифікацію (як правило, з підтвердженою автентифікацією), якщо він може перевірити, що переданий input являє собою дійсний principal.

2.Отримати AuthenticationException, якщо він вважає, що переданий input являє собою не дійсний principal.

3.Повернути null, якщо він не може прийняти рішення.

Різновидів автентифікації дуже багато, тому найпоширенішою реалізацією AuthenticationManager є ProviderManager, який делегує до ланцюга екземплярів інших інтерфейсів AuthenticationProvider [7]. AuthenticationProvider трохи схожий на AuthenticationManager, але він має додатковий метод, щоб дозволити абоненту запитувати, чи підтримує він даний тип автентифікації:

```
public interface AuthenticationProvider {

        Authentication authenticate(Authentication authentication)
                        throws AuthenticationException;

        boolean supports(Class<?> authentication);

}
```

Рисунок 2.6 – Інтерфейс AuthenticationProvider

ProviderManager може підтримувати декілька різних механізмів автентифікації в одній програмі, делегуючи до ланцюга інтерфейсів AuthenticationProviders [8]. Якщо ProviderManager не розпізнає певний тип екземпляра Authentication, то він буде пропущений.

У ProviderManager має опціонального предка, з яким він може обмінюватися інформацією у випадку, якщо всі провайдери повернуть null. Якщо предок недоступний, то повернений null призводить до AuthenticationException.

Іноді програма містить логічні групи захищених ресурсів (наприклад, всі веб-ресурси, які відповідають шаблону шляху / api / **), і кожна група може мати власний спеціалізований AuthenticationManager. Часто кожен із них є класом ProviderManager, і вони ділять предка. Тоді предок є своєрідним "глобальним" ресурсом, який виступає як резерв для всіх постачальників.

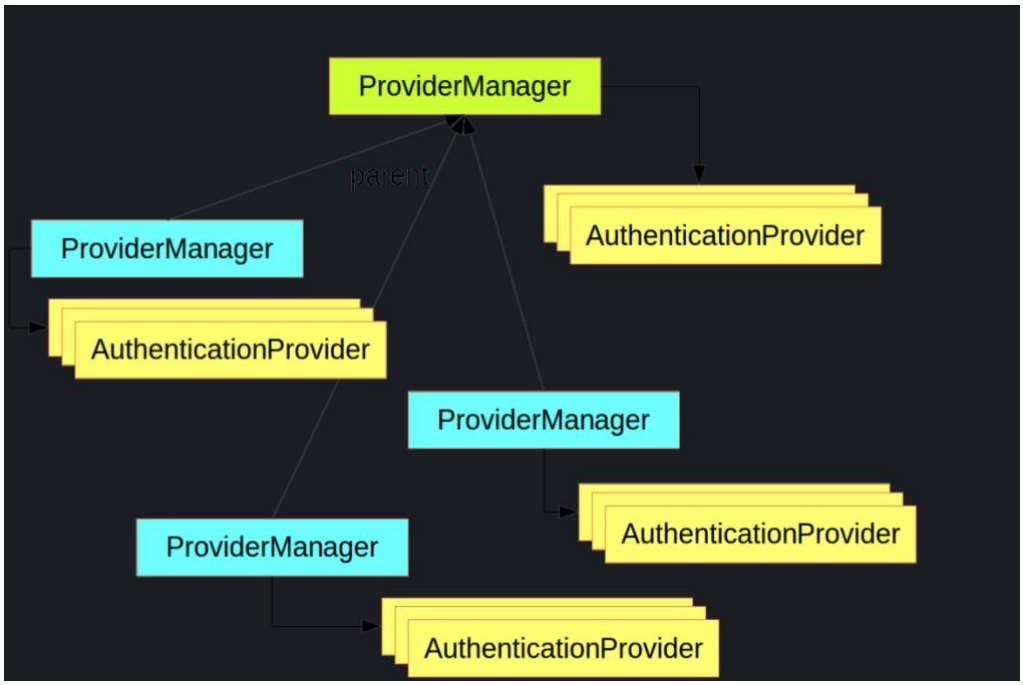


Рисунок 2.7 – Схема взаємодії компонентів які відповідають за автентифікацію.

Web Security

Клієнт надсилає запит до програми, і контейнер вирішує, які фільтри та сервлет застосовуються до нього на основі шляху URI-запиту. Щонайбільше один сервлет може обробляти один запит, але фільтри утворюють ланцюг, тому вони впорядковані, і насправді фільтр може накласти вето на решту ланцюга, якщо він хоче обробити сам запит. Фільтр також може змінювати запит та відповідь, який використовується у фільтрах нижнього потоку та сервлеті. Порядок ланцюга фільтрів дуже важливий, і Spring Boot керує ним за допомогою двох механізмів: один полягає в тому, що бін типу (Filter) може мати анотацію @Order або реалізувати Orders, а інший - вони можуть бути частиною FilterRegistrationBean, котрий сам має Order як частину свого API [8]. Деякі нестандартні фільтри визначають власні константи, які допомагають сигналізувати про те, який порядок вони мають мати відносно один одного.

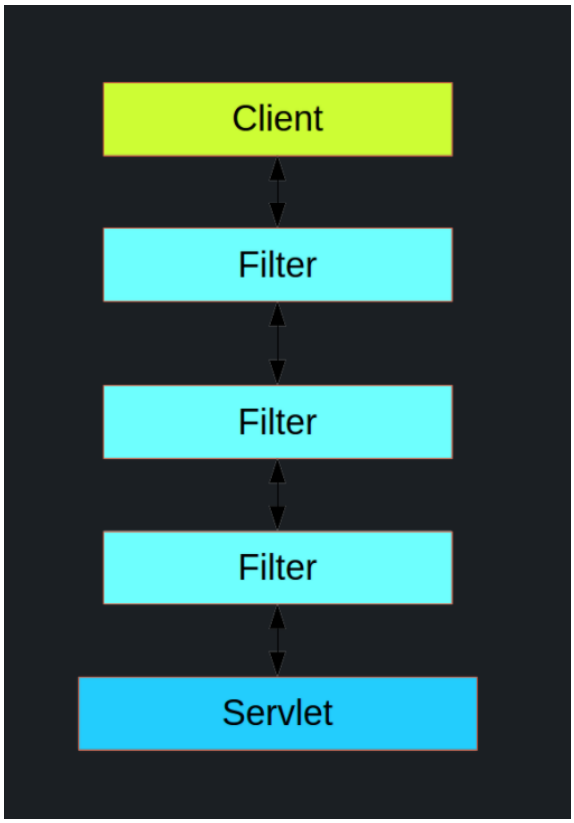


Рисунок 2.8 – Схема проходження запиту крізь фільтри.

2.1.5 Maven

Maven - це фреймворк для автоматизації збірки проектів, компіляцій, тестування, створення Jar-файлів і генерації документації.

Якщо складати великі проекти з командного рядка, то команда для складання буде дуже довгою і її іноді записують у скрипт. Але скрипти залежать від платформи, команди відрізняються для різних операційних систем. Для того, щоб спростити цей процес використовуються інструменти для складання проектів. Maven забезпечує розробнику декларативне складання проекту. У файлі POM.xml міститься його декларативний опис, а не окремі команди [4]. Усі завдання з обробки файлів Maven працюють через плагіни.

Рисунок 2.9 – Логотип Maven

Основні переваги Maven:

- Незалежність від операційної системи. Файл проекту один і той же;
- Дає змогу керувати залежностями і легко вирішувати конфлікти версій при переході на нові версії бібліотек;
- Можливість збірки з командного рядка (Continuous Integration);
- Добре налаштована інтеграція з середовищами розробки. При цьому більше нічого налаштовувати не потрібно, проект одразу готовий до подальшої роботи;
- А з попереднього пункту випливає, що якщо проекти розробляються в різних середовищах розробки, то це хороший спосіб передачі налаштувань;
- Декларативний опис проекту.

Основні недоліки Maven:

- Високий поріг входження для початку роботи з Maven;
- Необхідність наявності доступу до Інтернету.

2.1.6 PostgreSQL

PostgreSQL оснащений багатьма функціями, спрямованими на допомогу розробникам у створенні програм для захисту цілісності даних та побудови

середовищ, стійких до відмов, та допомагають керувати даними незалежно від того, наскільки великий чи малий набір даних. Окрім того, PostgreSQL є безкоштовним та з відкритим кодом. Розробник можете визначати власні типи даних, створювати власні функції, навіть писати код з різних мов програмування, не перекомпільовуючи свою базу даних.

Розроники PostgreSQL намагаються відповідати стандарту SQL, коли така відповідність не суперечить традиційним особливостям або може призвести до поганих архітектурних рішень. Багато функцій, необхідних стандарту SQL, підтримуються, хоча іноді мають дещо різний синтаксис або функції.

Подальші кроки до відповідності можна очікувати з часом. Станом на випуск версії 12 у жовтні 2019 року, PostgreSQL відповідає щонайменше 160 із 179 обов’язкових функцій для відповідності Core SQL: 2016. Станом на час написання цього тексту жодна реляційна база даних не відповідає цьому стандарту.

Мережеві адреси у PostgreSQL

PostgreSQL забезпечує зберігання різних типів мережевих адрес. Тип даних CIDR (Classless Internet Domain Routing) відповідає конвенції щодо мережевих адрес IPv4 та IPv6 [2]. Деякі приклади для CIDR:

192.168.100.128/25

10.1.2.3/32

2001: 4f8: 3: ba: 2e0: 81ff: fe22: d1f1 / 128

:: ffff: 1.2.3.0/128

Для зберігання мережевих адрес також доступний тип даних INET, який використовується для хостів IPv4 та IPv6, де підмережа не є обов'язковою [3]. Тип даних MACADDR може використовуватися для зберігання MAC-адрес для ідентифікації обладнання, таких як 08-00-2b-01-02-03.

MySQL та MariaDB мають деякі функції INET для перетворення мережевих адрес, але не надають типи даних для власного зберігання мережевих адрес. Firebird також не має типів мережевої адреси.

Багатовимірні масиви

Оскільки PostgreSQL є об'єктно-реляційною базою даних, масиви значень можуть зберігатися для більшості наявних типів даних. Зробити це можна, додавши квадратні дужки до специфікації типу даних для стовпчика або використовуючи вираз ARRAY. Розмір масиву можна вказати, але це не потрібно. Розглянемо приклад для демонстрації використання масивів:

```
 -- create a table where the values are arrays
CREATE TABLE holiday_picnic (
     holiday varchar(50) -- single value
     sandwich text[], -- array
     side text[] [], -- multi-dimensional array
     dessert text ARRAY, -- array
     beverage text ARRAY[4] -- array of 4 items
);

 -- insert array values into the table
INSERT INTO holiday_picnic VALUES
     ('Labor Day',
     '{"roast beef","veggie","turkey"}',
     '{
        {"potato salad","green salad","macaroni salad"},
        {"chips","crackers"}
     }',
     '{"fruit cocktail","berry pie","ice cream"}',
     '{"soda","juice","beer","water"}'
     );
```

Рисунок 2.10 – Демонстрація використання масивів у PostgreSQL.

MySQL, MariaDB та Firebird не мають такої можливості. Для зберігання масиву таких значень у традиційній реляційній базі даних потрібна окрема таблиця з рядком для кожного зі значень масиву.

Підтримка JSON

Підтримка JSON PostgreSQL дає змогу зберігати дані з можливістю індексуванням. Це може бути корисно, коли структура даних вимагає певної гнучкості, оскільки вона все ще змінюється в процесі розробки або коли невідомо, які поля даних буде містити об'єкт даних.

Тип даних JSON застосовує справжній JSON, який дає змогу потім використовувати спеціалізовані оператори JSON та функції, вбудовані в PostgreSQL, для запитів та обробки даних. Також доступний тип JSONB - двійкова форма JSON, де видаляються пробіли, порядок об’єктів не зберігається, але замість цього зберігається оптимально, і зберігається лише останнє значення для дублюючих ключів. JSONB, як правило, є кращим форматом, оскільки він вимагає менше місця на об'єкт, може бути індексований і може бути оброблений швидше, оскільки він не вимагає повторного аналізу.

У MySQL 5.7.8 та MariaDB 10.0.1 була представлена підтримка корінних об'єктів JSON. Хоча для JSON існують різноманітні функції та оператори, які зараз доступні в цих базах даних, вони не піддаються індексації способом JSONB як в PostgreSQL. Firebird ще не не володіє таким функціоналом і лише підтримує JSON-об’єкти як текст.

Створення нового типу даних

Якщо набір існуючих типів даних PostgreSQL був недостатній, розробник може використовувати команду CREATE TYPE для створення нових типів.. Ось приклад створення та запиту нового складеного типу:

```
 -- create a new composite type called "wine"
CREATE TYPE wine AS (
     wine_vineyard varchar(50),
     wine_type varchar(50),
     wine_year int
);

 -- create a table that uses the composite type "wine"
CREATE TABLE pairings (
     menu_entree varchar(50),
     wine_pairing wine
);

 -- insert data into the table using the ROW expression
INSERT INTO pairings VALUES
     ('Lobster Tail',ROW('Stag''s Leap','Chardonnay', 2012)),
     ('Elk Medallions',ROW('Rombauer','Cabernet Sauvignon',2012));
```

Рисунок 2.11 – Демонстрація створення нового типу даних.

Цілісність даних

Розробники PostgreSQL прагнуть відповідати стандарту ANSI-SQL: 2008, повністю сумісний з ACID і добре відомий своєю твердою референтною та транзакційною цілісністю. Первинні ключі, обмежувальні та каскадні зовнішні ключі, унікальні обмеження, non-null обмеження, контрольні обмеження та інші функції цілісності даних забезпечують збереження лише перевірених даних.

2.1.7 Spring Data

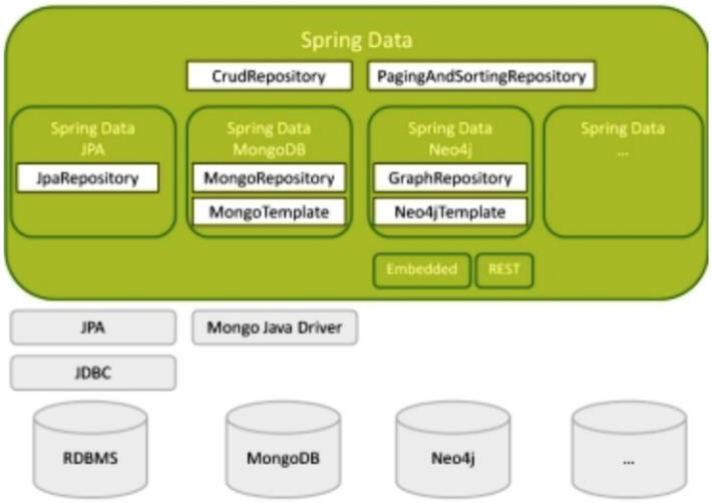


Рисунок 2.12 – Схема Spring Data

Spring Data JPA Repositories

Побудова репозиторію полягає у визначенні договору, який буде реалізовувати рівень доступу до даних. Цей контракт потім може бути включений і зв'язаний клієнтським кодом, який потребує доступу до даних визначеним розробником чином. Це насправді означає, що сховище Spring - це реалізація шаблону Data Access Object.

Всі Spring Data JPA репозиторії - це Java інтерфейси замість класів. Ці інтерфейси пов'язані з об'єктом JPA [1]. Кожне сховище JPA може виконувати операції з доступу до даних лише для цього конкретного об'єкта та його атрибутів даних. Це допомагає побудувати доступу до сховище JPA на договорі DAO для цієї сутності та його резервних даних.

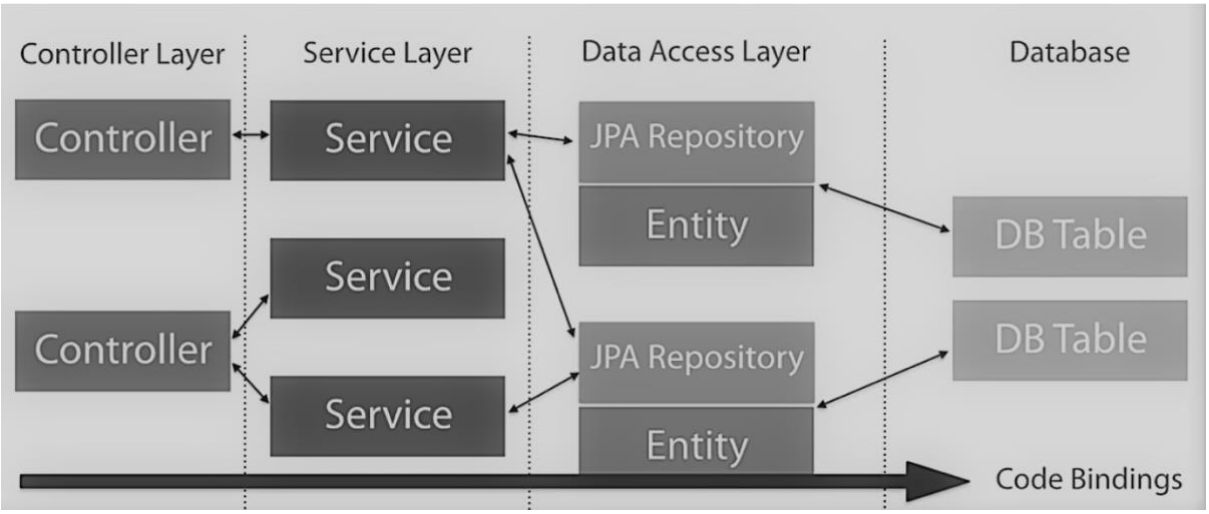


Рисунок 2.13 – Схема роботи сервера.

Функціональність:

- Query DSL;
- CRUD operations;
- Paging and sorting;
- Helpers;
- count();
- exists(Long id);
- flush();
- deleteInBatch(Iterable entities);

Також є можливості сортування, і, нарешті, сховище JPA містить кілька помічників, завдяки яким робота з кодом який відповідає за доступ до даних значно полегшується. Деякі з них включають пошук підрахунку резервної таблиці БД, тестування того, чи існує запис у базі даних, flush змін у стійкий контекст (persistence context) до бази даних та обробку видалення декількох об'єктів за допомогою одного запиту за допомогою зручного методу deleteInBatch ().

ВИСНОВОК ДО РОЗДІЛУ 2

В цьому розділі було описано архітектурні особливості вибраного стеку технологій та їх переваг, які завдяки своєму функціоналу підходять для реалізації даного проекту.

Проаналізовані деталі всіх частини системи і створена характеристика її функціоналу. Було виконано опис фреймворка Spring та визначені його основні складових, а саме (Spring Boot, Spring Data та Spring MVC). Цей фреймворк є найєфективнішим для побудови сучасних серверних програм, які написані на мові програмування Java. Системою збірки проекту був обраний Maven, а в якості провайдера бази даних та системи для взаємодії з нею було обрано PostgreSQL та Spring.

В результаті проведеного дослідження було вирішено побудувати архітектуру системи на основі шаблону MVC і архітектури сервера CSR.
Перевагу саме цьому шаблону та архітектурі було надано за їх гнучкість і перевірену часом простоту.

РОЗДІЛ 3.ОПИС ВНУТРІШНЬОЇ СТРУКТУРИ СЕРВЕРА

2.1 Опис програмних модулів системи

№ з/п	Позначення	Призначення
1	UserController.java	Контролер, який відповідає за реєстрацію, редагування, оновлення та видалення користувача.
2	MailParamsController.java	Контролер, який відповідає за редагування поштових налашутвань
3	ReceivingTemplateController.java	Контролер, який відповідає за отримання шаблонів
4	SendingTemplateController.java	Котролер, який відповідає за надсиланя шаблонів
5	SendMailController.java	Контролер, який відповідає за надсилання електронних листів
6	FriendController.java	Контролер, який відповідає за реєстрацію, редагування, оновлення та видалення адресантів.
7	MessageTemplateController.java	Контролер, який відповідає за реєстрацію, редагування, оновлення та видалення шаблонів повідомлень.
8	EmailReceivingServiceImpl.java	Сервіс, який відповідає за бізнес-логіку для отримання електронних листів
9	EmailSendingServiceImpl.java	Сервіс, який відповідає за бізнес-логіку для надсилання електронних листів

№ з/п	Позначення	Призначення
10	FriendServiceImpl.java	Сервіс, який відповідає за бізнес-логіку для редагування даних адресантів
11	MailParamsServiceImpl.java	Сервіс, який відповідає за бізнес-логіку для редагування поштових налаштувань
12	ReceivingTemplateServiceImpl.java	Сервіс, який відповідає за бізнес-логіку для отримання шаблонів
13	SendingTemplateServiceImpl.java	Сервіс, який відповідає за бізнес-логіку для надсиланя шаблонів
14	UserServiceImpl.java	Сервіс, який відповідає за бізнес-логіку для редагування даних користувача
15	MessageTemplateService.java	Сервіс, який відповідає за бізнес-логіку для редагування шаблонів повідомлень

2.2 Опис основних методів в класах

Назва метода	Аргументи	Значення, що повертається	Призначення методу
findAllUsers	-	List<UserDto> users	Отримати список всіх користувачів
getUserById	Long id	ResponseEntity<UserDto> responseEntity	Отримати користувача за його id
saveUser	UserDto userDTO	ResponseEntity<Void> responseEntity	Зберегти нового користувача
updateUser	UserDto userDTO	ResponseEntity<Void> responseEntity	Оновити інформацію наявного користувача
deleteUser	Lond Id	ResponseEntity<Void> responseEntity	Видалити користувача
getAllReceivingTemplates	-	List<ReceivingTemplateDto> dtoList	Отримати всі шаблони для отримання
createReceivingTemplate	ReceivingTemplateDto receivingTemplateDto	ResponseEntity<ReceivingTemplateDto> responseEntity	Створити шаблон для отримання

Назва метода	Аргументи	Значення, що повертається	Призначення методу
deleteReceivingTemplateByRid	String rid	ResponseEntity<ReceivingTemplateDto> responseEntity	Видалити шаблон для отримання за Id
getReceivingTemplateByRid	String rid	ResponseEntity<ReceivingTemplateDto> responseEntity	Отримати шаблон для отримання за Id
updateReceivingTemplateByRid	String rid, ReceivingTemplateDto receivingTemplateDto	ResponseEntity<ReceivingTemplateDto> responseEntity	Оновити шаблон отримання за Id
getAllAccounts	-	List<MailParamsDTO> mailParamsDTOList	Отримати всі облікові записи
createMailAccount	MailParamsDTO mailParamsDTO	ResponseEntity<MailParamsDTO> responseEntity	Створити обліковий запис
getMessageTemplateByName	String name	ResponseEntity<MessageTemplateDto>	Отримати шаблон повідомлення за назвою

Назва метода	Аргументи	Значення, що повертається	Призначення методу
deleteMailAccountByRid	String rid	ResponseEntity<MailParamsDTO> responseEntity	Видалити обліковий запис
getMailAccountByRid	String rid	ResponseEntity<MailParamsDTO> responseEntity	Отримати обліковий записз за Id
updateMailAccountByRid	String rid, MailParamsDTO mailParamsDTO	ResponseEntity<MailParamsDTO> responseEntity	Оновити обліковий запис за Id
getAllSendingTemplates	-	List<SendingTemplateDto> sendingTemplateDtoList	Отримати всі шаблони для надсилання повідомлень
createSendingTemplate	SendingTemplateDto sendingTemplateDto	ResponseEntity<SendingTemplateDto> responseEntity	Створити шаблон надсилання

Назва метода	Аргументи	Значення, що повертається	Призначення методу
deleteSendingTemplateByRid	String rid	ResponseEntity<SendingTemplateDto> responseEntity	Видалення шаблону для відправки повідомлень за Id
getSendingTemplateByRid	String rid	ResponseEntity<SendingTemplateDto> responseEntity	Отримання шаблону для відправки повідомлень за Id
updateSendingTemplateByRid	String rid, SendingTemplateDto sendingTemplateDto	ResponseEntity<SendingTemplateDto> responseEntity	Оновлення шаблону для відправки повідомлень за Id
receiveRecentMessages	-	void	Отримати повідомлення які надійшли за останній день
updateMessageTemplate	MessageTemplateDto messageTemplateDto	ResponseEntity<Void> responseEntity	Оновити шаблон повідомлення

Назва метода	Аргументи	Значення, що повертається	Призначення методу
getMailStore	-	Store store	Отримати всі повідомлення з облікового запису
getMessageInfo	Message message	Void	Отримати інформацію про повідомлення
saveReceivedMessageToDB	List<String> addresses, String urlFromFileStorage, String messageContent	Void	Зберегти отримане повідомлення в базу даних
getMessageText	BodyPart part	String text	Отримати текст повідомлення
sendSimpleMessage	-	Void	Надіслати повідомлення

Назва метода	Аргументи	Значення, що повертається	Призначення методу
sendSimpleMessage	-	Void	Надіслати повідомлення
createTextOfMessage	String template, Map<String, String> params	String text	Сворити текст повідомлення
createJavaMailSender	-	JavaMailSender javaMailSender	Створити об’єкт класу JavaMailSender
responseEntity	T dto, HttpStatus httpStatus	<T> ResponseEntity<T> responseEntity	Створення об’єкта класу ResponseEntity за переданим DTO та статусом http
saveMessageTemplate	MessageTemplateDto messageTemplateDto	ResponseEntity<Void> responseEntity	Зберегти шаблон повідомлення

Назва метода	Аргументи	Значення, що повертається	Призначення методу
responseEntity	T dto, HttpStatus httpStatus	<T> ResponseEntity<T> responseEntity	Створення об’єкта класу ResponseEntity за переданим DTO та статусом http
responseEntity	HttpStatus httpStatus	<T> ResponseEntity<T> responseEntity	Створення об’єкта класу ResponseEntity за переданим статусом http
responseEntityTer	T dto	<T> ResponseEntity<T> responseEntity	Створення об’єкта класу ResponseEntity за переданим DTO
deleteMessageTemplateByName	String name	ResponseEntity<Void> responseEntity	Видалити шаблон повідомлення за назвою
findByTemplateName	String name	MessageTemplateEntity messageTemplateEntity	Знайти в базі даних шаблон повідомлення за назвою

2.3 Структура бази даних

База даних складається з таких таблиць:

- user_table – таблиця зберігання даних зареєстрованого користувача;
- friend_table – таблиця для зберігання доданих адресантів;
- message_table – таблиця для зберігання повідомлень отриманих користувачем від адресантів;
- message_template_table – таблиця для зберігання доданих користувачем шаблонів повідомлень;
- word_table – таблиця для зберігання доданих користувачем ключових слів для аналізу отриманих повідомлень від адресантів;
- date_table – таблиця для зберігання основних дат користувача.

ВИСНОВОК ДО РОЗДІЛУ 3

В даному розділі було описано основні модулі сервера, які використовуються для роботи системи створення, редагування та видалення користувачів та адресантів та шаблонів. Та основні модулі які приймають участь у бізнес-логіці для аналізу та формування повідомлень.

В результаті цього розділу ми отримали чітку структуру основних модулів архітертури сервера та визначили функціонал кожного модуля.

РОЗДІЛ 4. ТЕСТУВАННЯ ТА ОГЛЯД СЕРВЕРНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.4 Огляд програмного забезпечення

Спочатку повинна відбутися реєстрація користувача або вхід в систему зі своїм паролем та логіном. У випадку вдалої реєстрації користувача буде отримано код 201 Created.

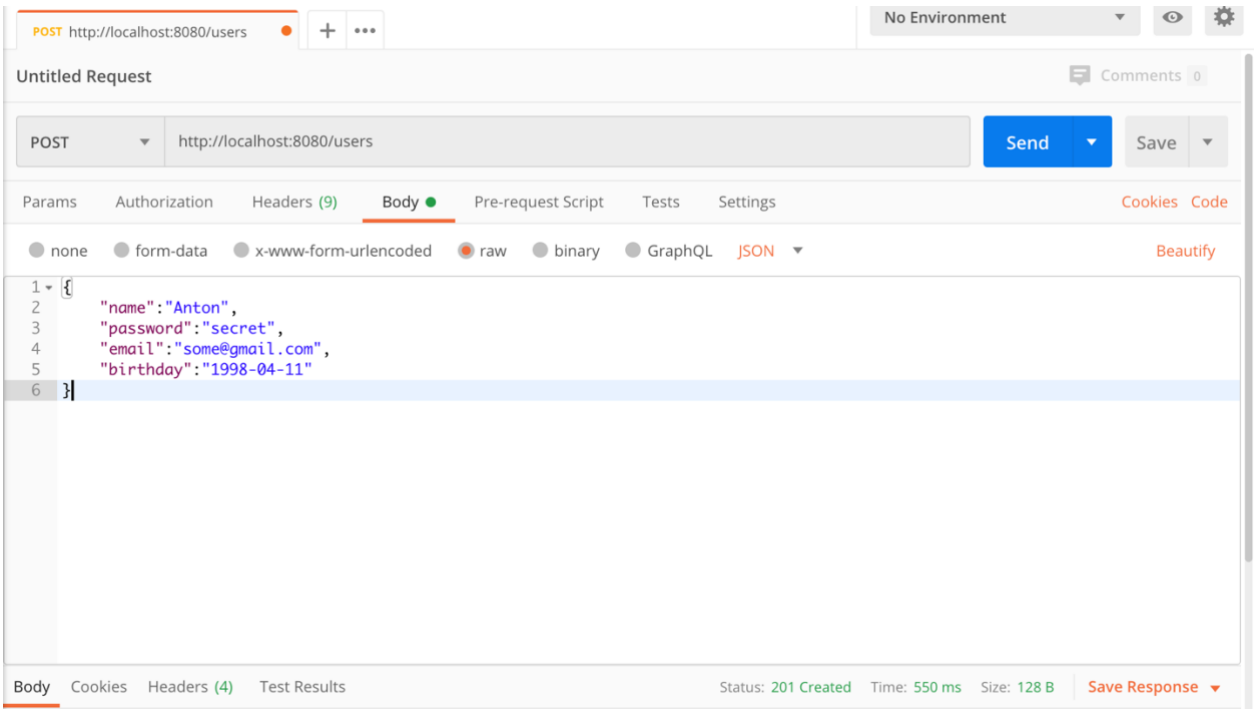


Рисунок 4.1 – Реєстрація користувача.

При спробі реєстрації за відстуності обов’язкого поля, буде код 400 Bad Request. На прикладі видно, що відсутнє поле email та у відповідь користувач отримує повідомлення: “must not be null” для поля email.

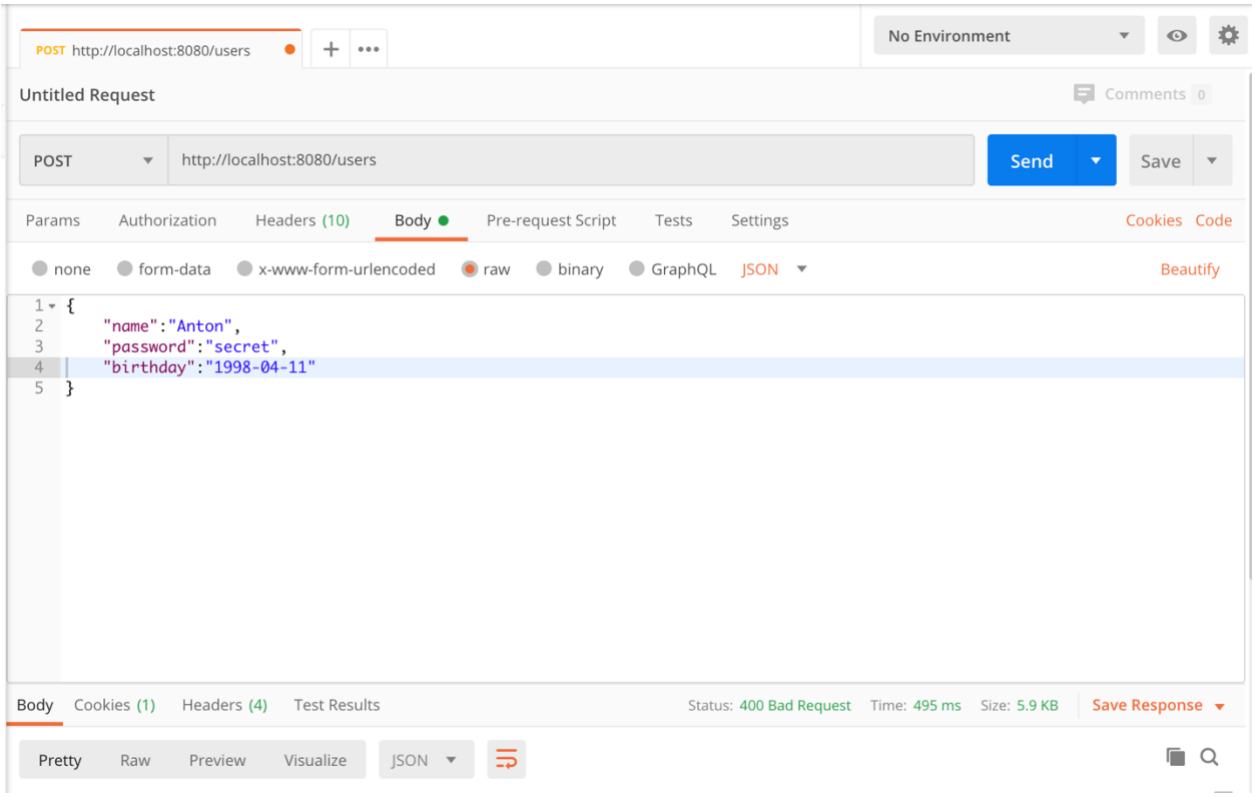


Рисунок 4.2- Спроба реєстрації користувача без електронної скриньки.

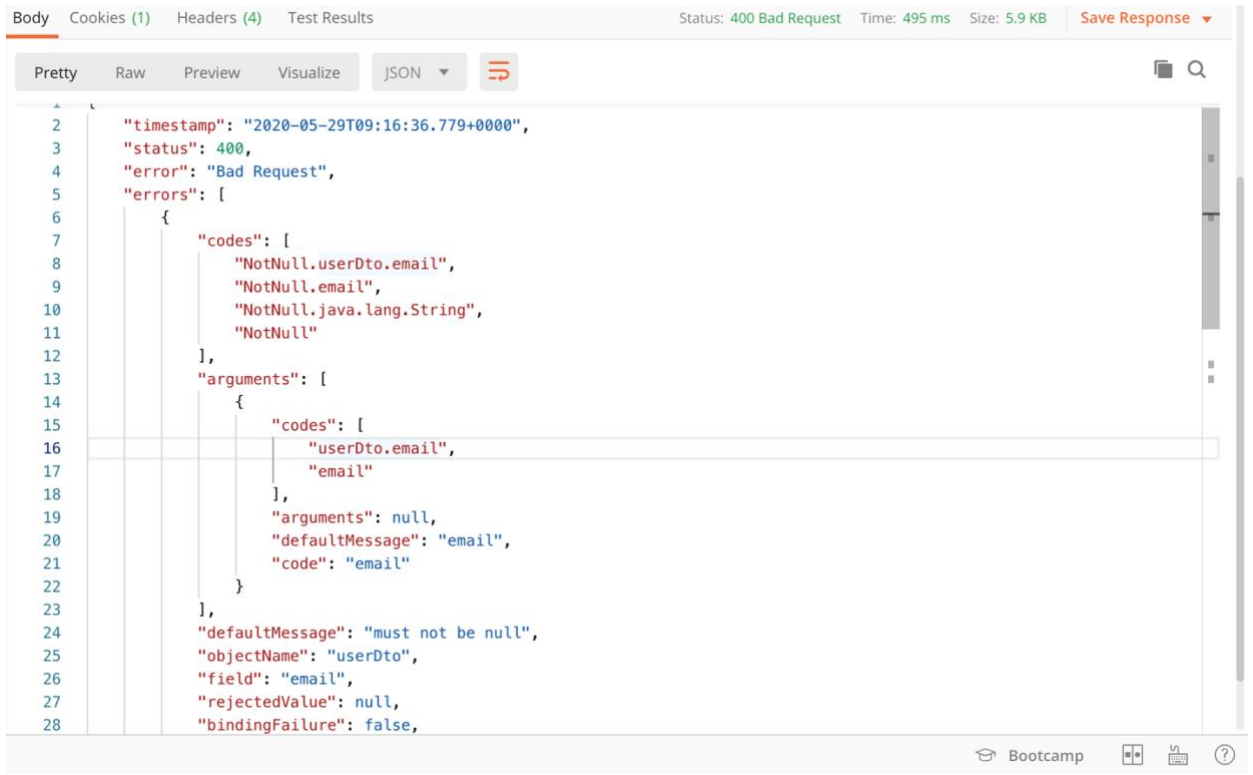


Рисунок 4.3- Спроба реєстрації користувача без електронної скриньки.

У випадку спроби отримати доступ без авторизації, користувач отримає код 401 Unauthorized.

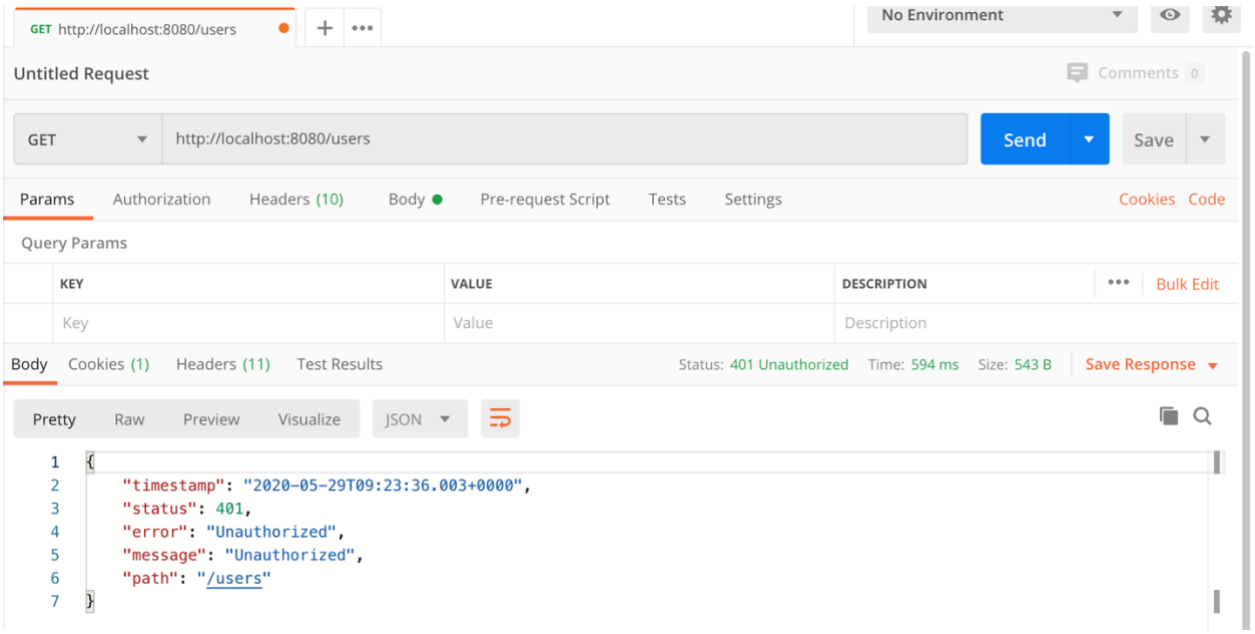


Рисунок 4.4 – Спроба запиту без авторизації.

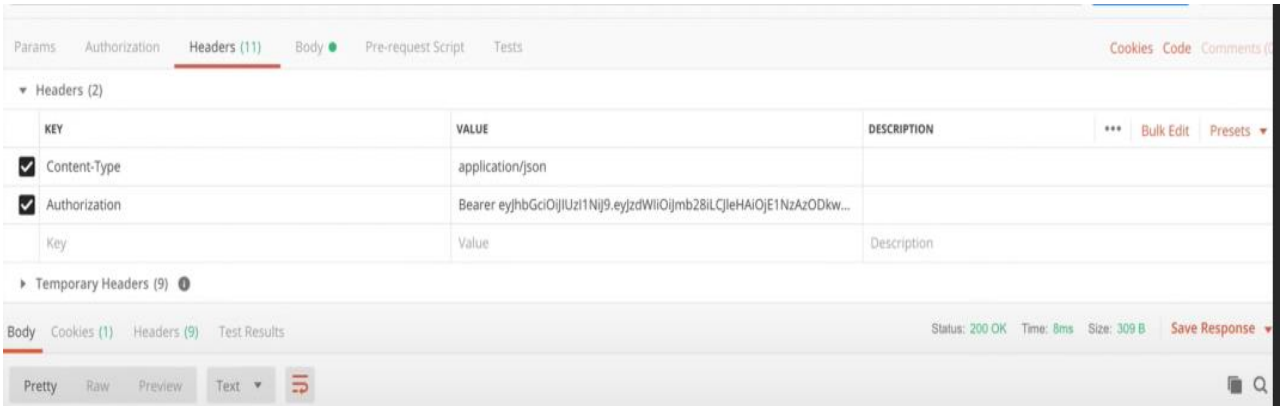


Рисунок 4.5- Успішно виконаний запит з авторизацією.

Після того як користувач зареєструвався або залогінився, він отримує доступ до наступного функціоналу. Тут буде розглянуте оновлення профілю користувача. У випадку вдалого оновлення профілю користувача буде отримано код 204 No Content.

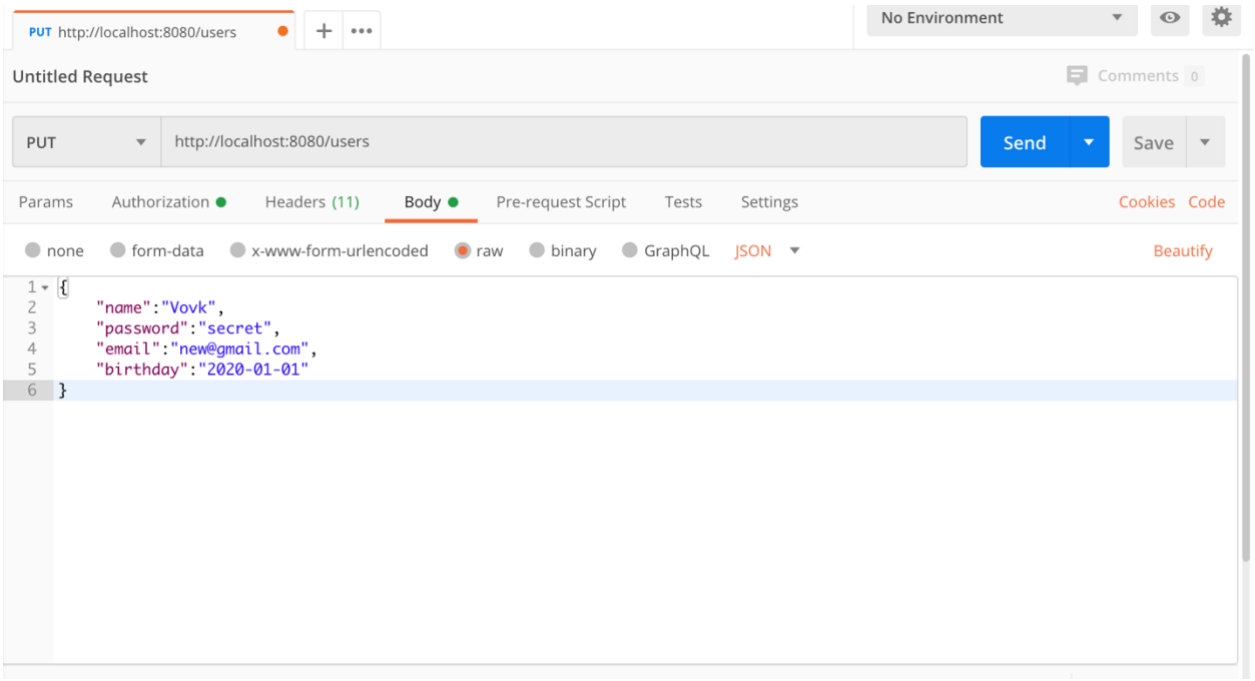


Рисунок 4.6 - Вдале оновлення профілю користувача.

За бажанням користувач може видалити свій профіль. При вдалому запиті на видалення користувач у відповідь отримає код 200 OK. При відсутності такого користувача в базі даних буде отримано код 500 Internal Server Error.

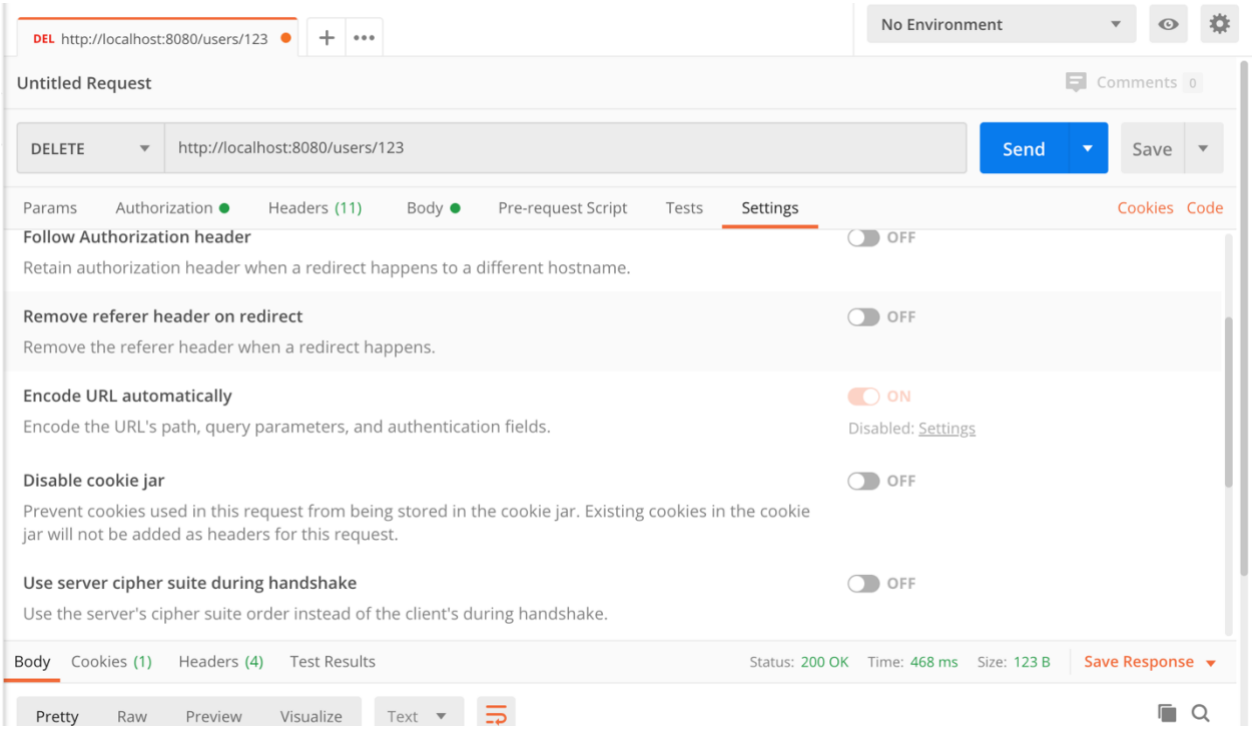


Рисунок 4.7 – Успішне видалення профілю користувача.

Після огляду створення та редагування профілю користувача буде розглянуто додавання одресантів в базу даних. При вдалому додаванні адресантів до бази даних буде отримано код 201 Created.

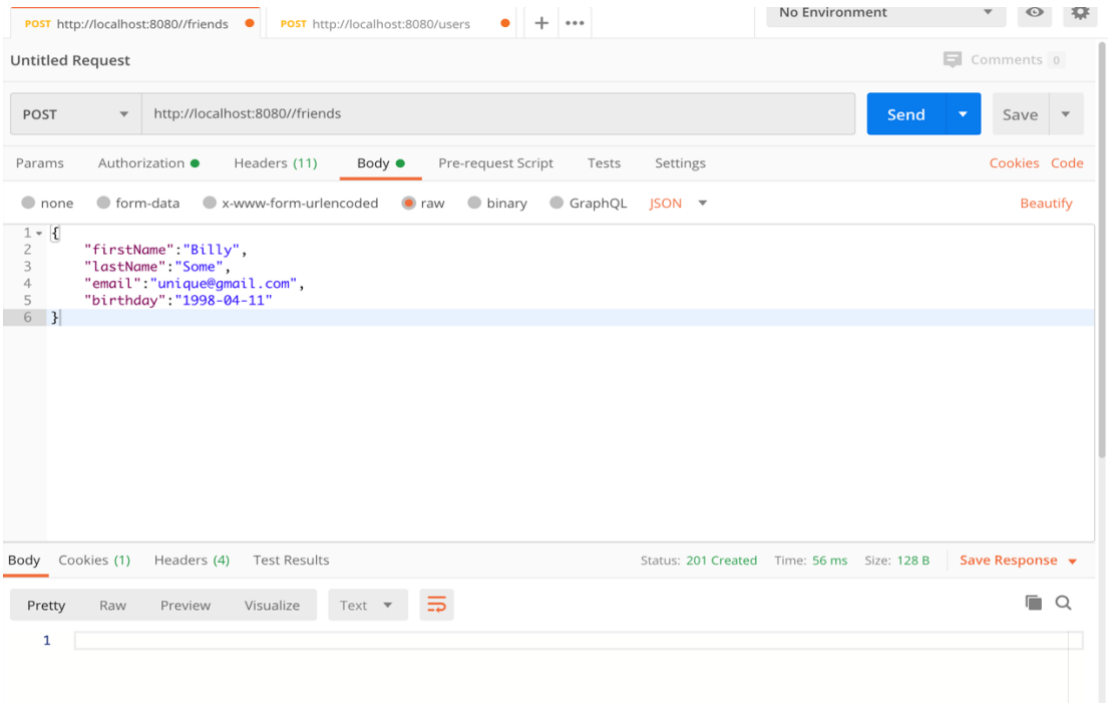


Рисунок 4.8 – Успішне додавання адресанта до бази даних

У випадку невдалого додавання користувача буде отримано код 400 BadRequest. Це розглянуто на прикладі відсутності одного з обов'язкових полей для формування DTO.

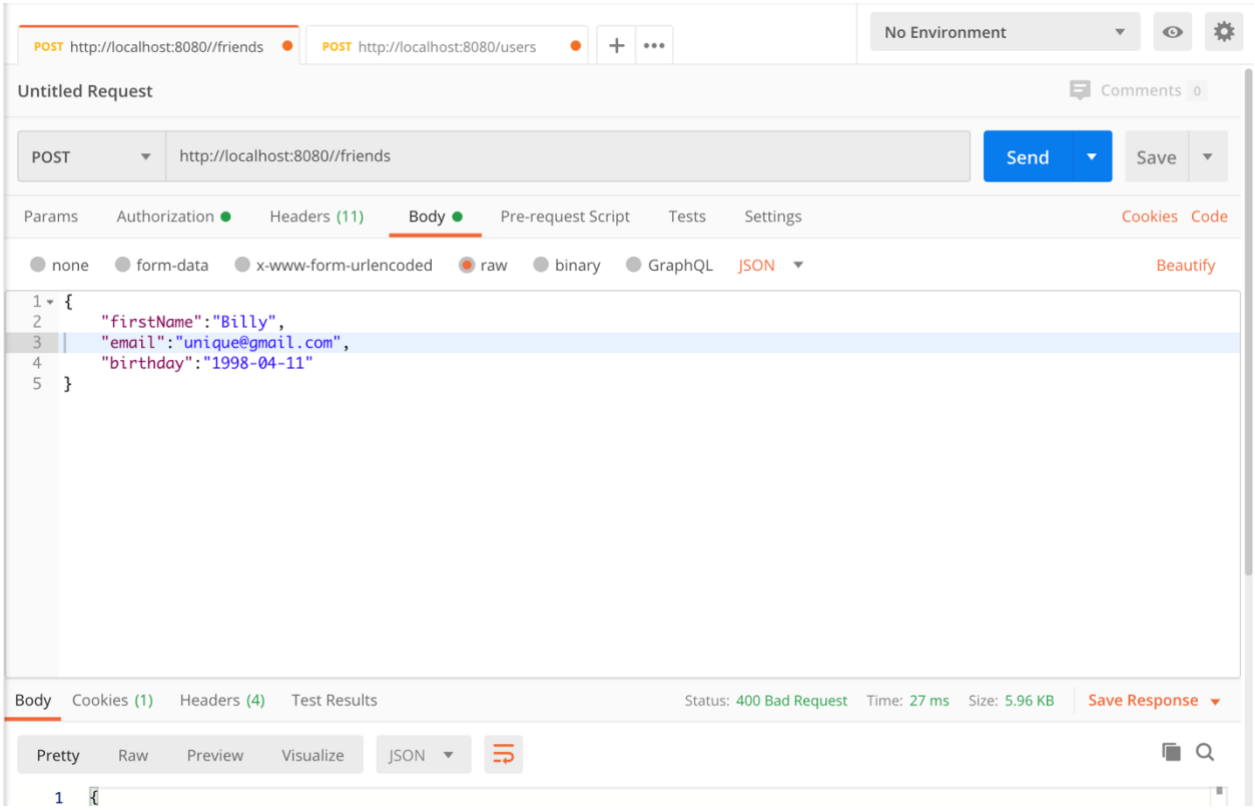


Рисунок 4.9 – Спроба додавання адресанта без всіх необхідних полей.

Відстунє поле lastName і у відповідь користувач отримає код 400 та повідомлення: "must not be null" для поля lastName.

```
{
    "timestamp": "2020-05-30T07:35:09.265+0000",
    "status": 400,
    "error": "Bad Request",
    "errors": [
        {
            "codes": [
                "NotNull.friendDto.lastName",
                "NotNull.lastName",
                "NotNull.java.lang.String",
                "NotNull"
            ],
            "arguments": [
                {
                    "codes": [
                        "friendDto.lastName",
                        "lastName"
                    ],
                    "arguments": null,
                    "defaultMessage": "lastName",
                    "code": "lastName"
                }
            ],
            "defaultMessage": "must not be null",
            "objectName": "friendDto",
            "field": "lastName",
            "rejectedValue": null,
            "bindingFailure": false,
            "code": "NotNull"
        }
```


Рисунок 4.10 – Спроба додавання адресанта без всіх необхідних полей.

Коли користувач робить запит на отримання всіх доданих адресантів, то він отримує список адресантів та код 200 OK.

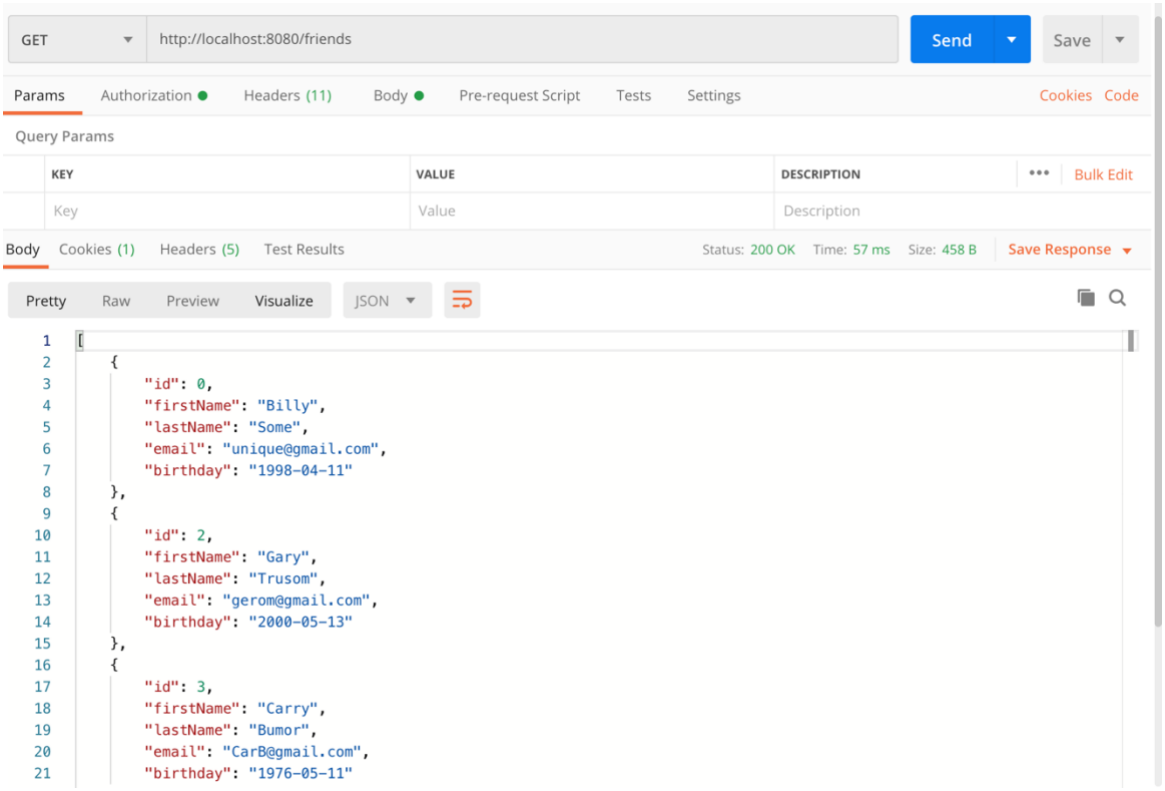


Рисунок 4.11 – Відповідь на запит на отримання всіх адресантів доданих користувачем.

За бажання користувач може редагувати та видаляти адресантів. У випадку вдалого редагування у відповідь буде отримании код 204 No Content. А при вдалому видаленні з бази даних буде отримано код 200 ОК.

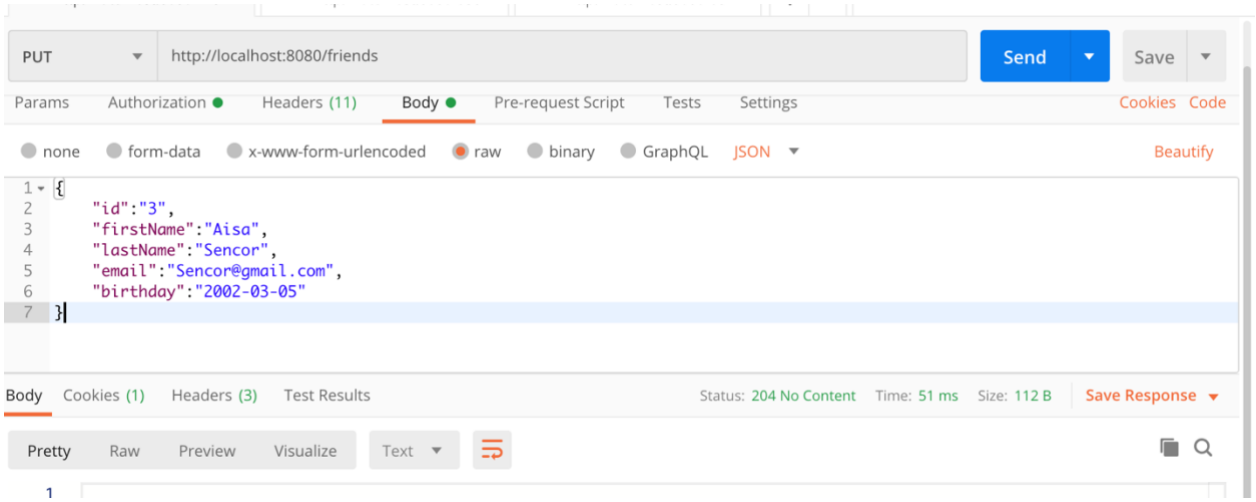


Рисунок 4.12 -Успішне редагування інформації про адресанта.

В цьому прикладі було зроблено спробу редагувати адресанта з невірним форматом електронної пошти. У відповідь було отримано код 404 Bad Request та повідомлення: “Email address has invalid format: Sencordvsdf.gmail.com”

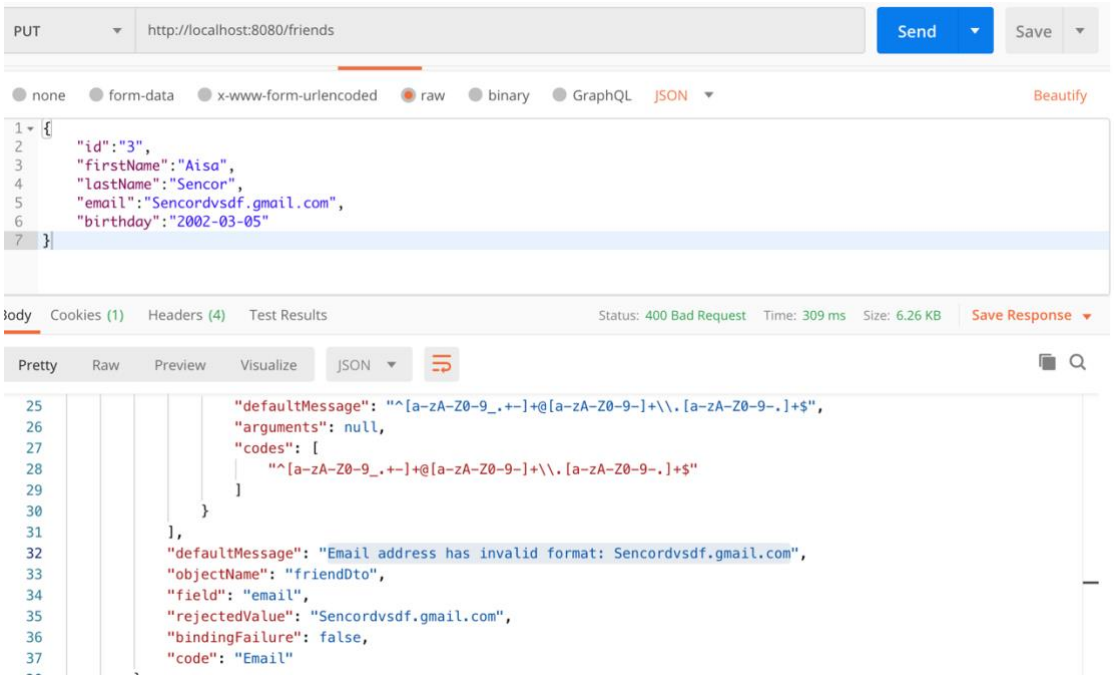


Рисунок 4.13 – Відповідь на спробу редагування адресанта з невірним форматом електронної пошти.

Для формування особистих повідомлень до адресантів користувач має надати шаблони які він вважає необхідними. Після чого відповідні шаблони будуть використані у відповідні дати.

Приклад додавання шаблону для відправки повідомлень. У випадку вдалого збереження буде отриманий код 201 Created.

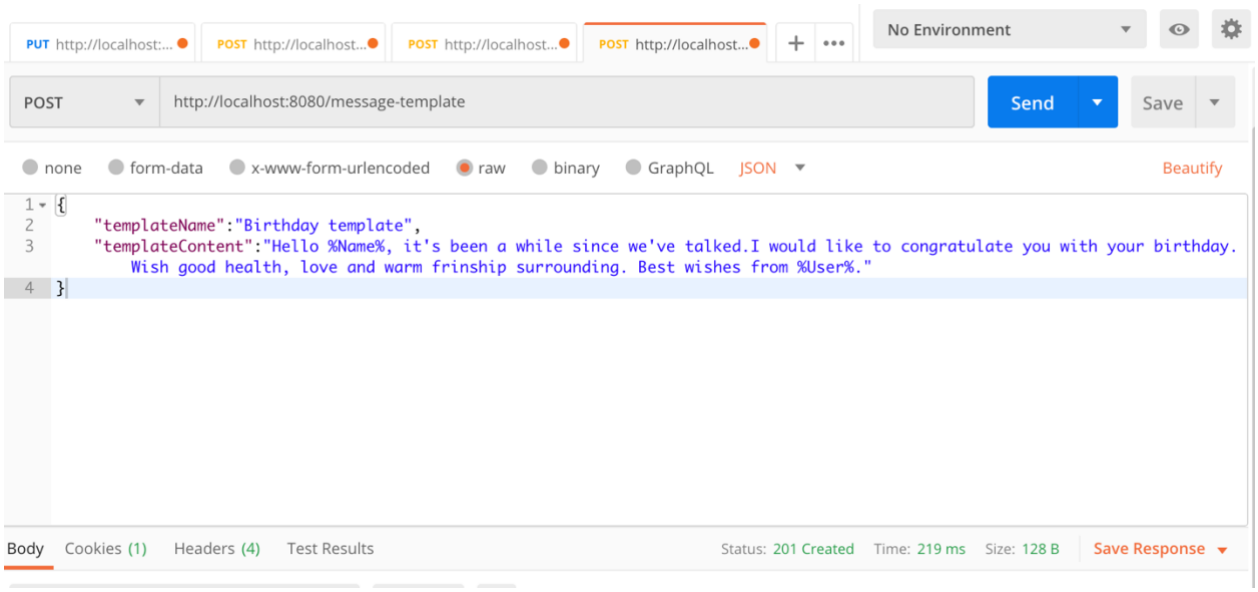


Рисунок 4.14 - Успішне додавання шаблону повідомлення.

При спробі додавання шаблону без всіх необхідних полей для формування DTO, користувач отримає код 400.

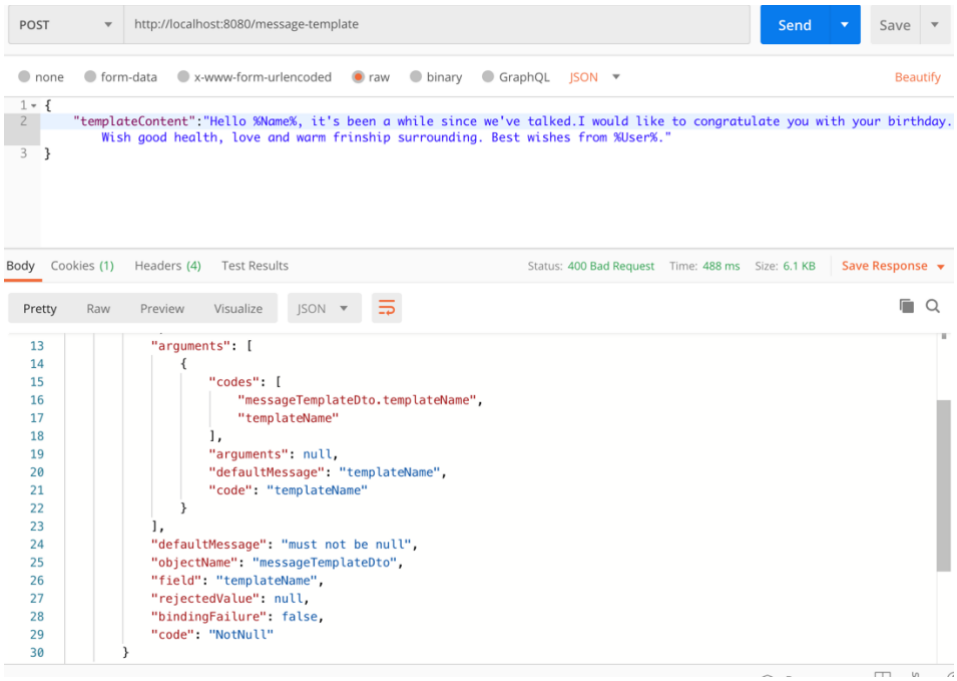


Рисунок 4.15 -Відповідь на невдалий запит створення шаблону повідомлення.

Було додано кілька адресантів для тестування системи аналізу повідомлень.

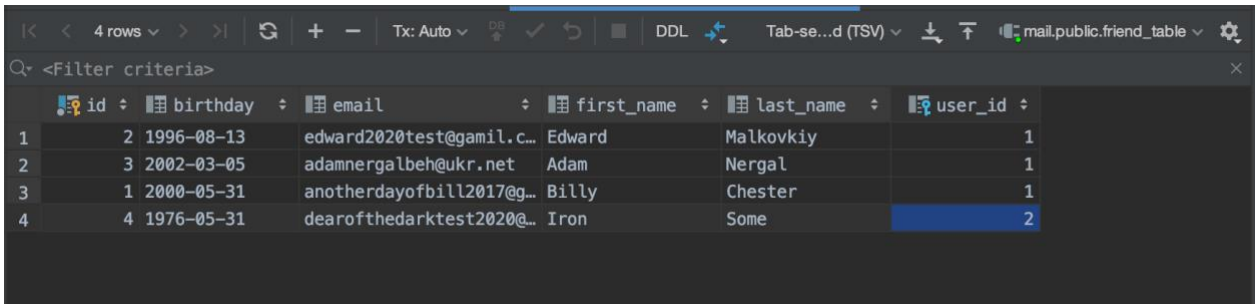


Рисунок 4.16 – Адресанти в базі даних.

Коли настає день з датою вказаною в базі даних, система робить перевірку на наявність листів від адресанта в той самий день минулого року та за день до і після. У разі знайденого повідомлення від адресанта, буде обраний потрібний шаблон та передані відповідні до адресанта дані для формування повідомлення та його надсилання.

В базі даних системі наявно два користувача, тестування проводиться з користувачем з id = 1.

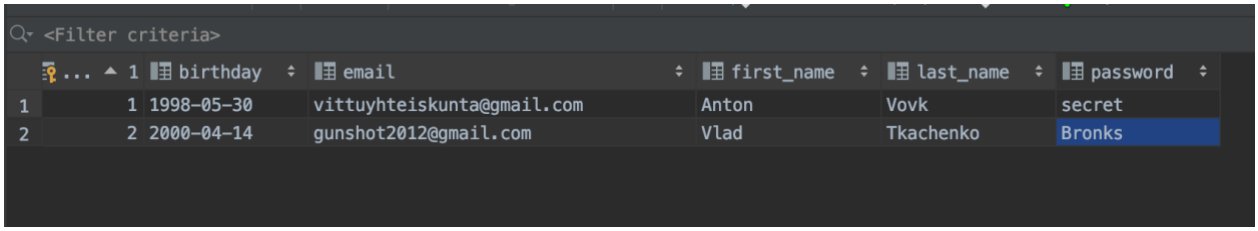


Рисунок 4.17 – Користувачі в базі даних.

Було створено повідомлення від імені адресанта з id = 1. Від адресанта з id = 4 повідомлення з привітанням надіслано не було. Для перевірки в обох адресантів дата народження вказана в один і той самий день.

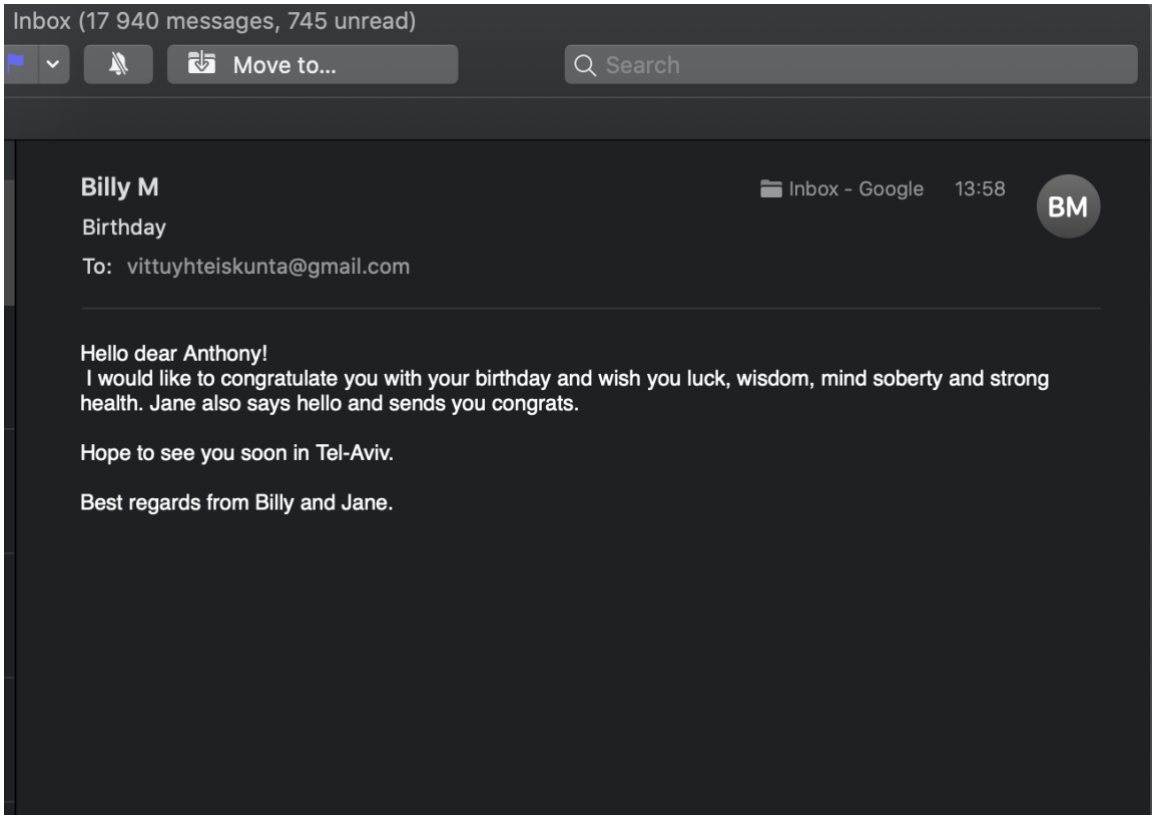


Рисунок 4.18 – Надіслане повідомлення з привітанням.

Наступного дня адерсантом з id = 1 було отримане привітальне повідомлення за доданим шаблоном. Користувач з id = 4 не отримав повідомлення з привітанням як і планувалося.

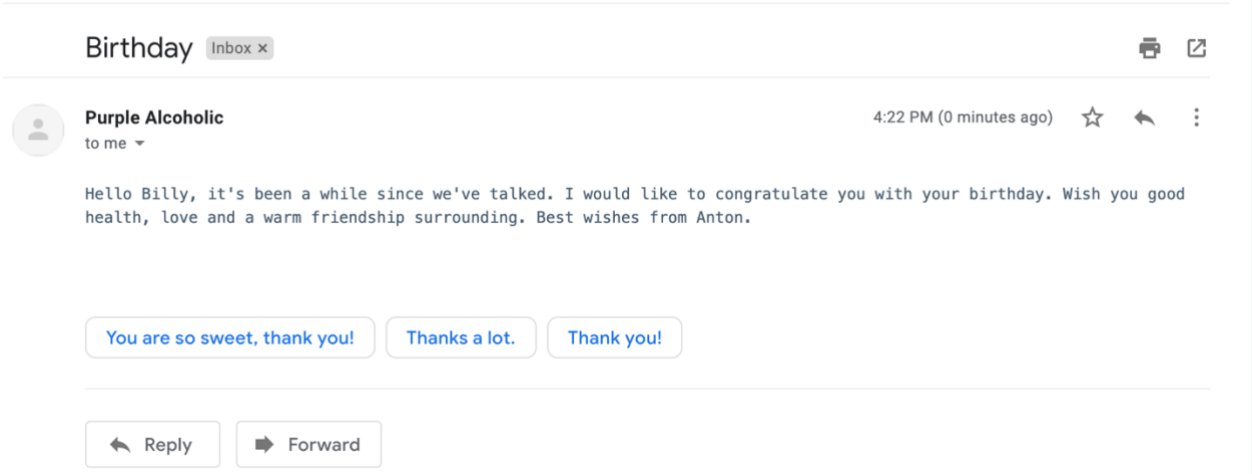


Рисунок 4.19 – Надіслане повідомлення автоматизованою системою.

Далі було перевірено надсилання привітань з іншими святами. Для цього було обрано день Києва та доданий відповідний шаблон.

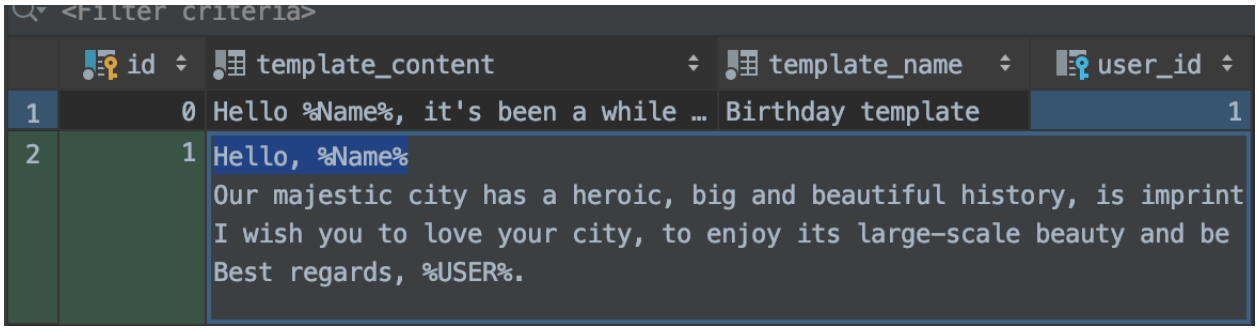


Рисунок 4.20 – Шаблон який було використано для привітання зі святом дня Києва.

В даному випадку система аналізу повідомлень було вимкнена. Абсолютно всі адресанти отримали повідомлення на свою електронну пошту.

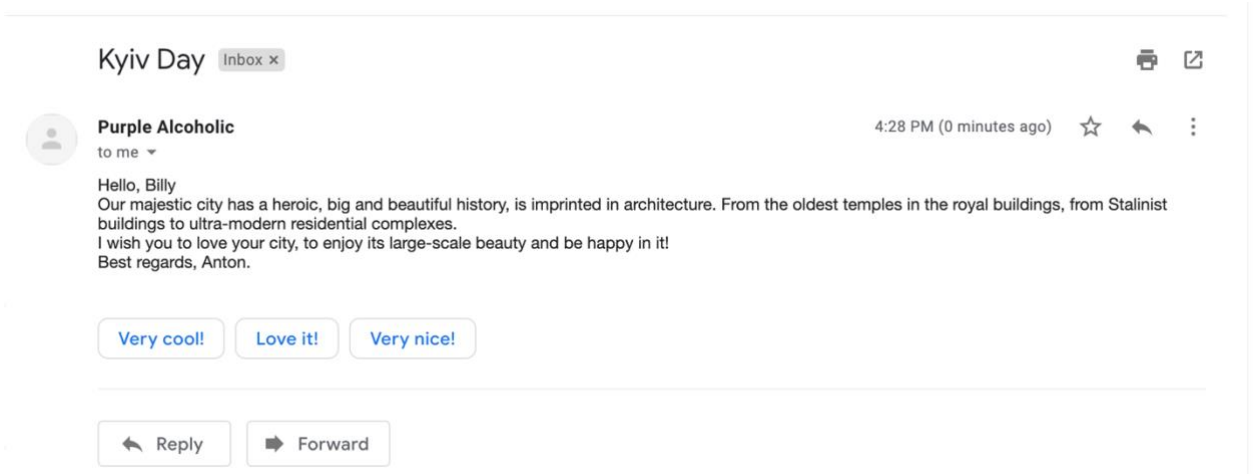


Рисунок 4.21 – Отримане повідомлення одним із адресантів.

ВИСНОВОК ДО РОЗДІЛУ 4

Було продемонстровано функціонал розробленого сервера, його можливостей та поведінки при введенні даних у правильному та неправильному форматі.

В ході проведення тестів було виявлено, що при введенні будь-яких даних сервер продовжує працювати та користувач може продовжувати роботу. При передачі серверу даних у невірному форматі, сервер надає відповідь у вигляді коду помилки та повідомлення, що дає змогу користувачу легко зрозуміти в чому помилка.

ВИСНОВКИ

В даному дипломному проекті було спроектовано та розроблено сервер який дозволяє виконувати автоматизований аналіз та формування повідомлень. У результаті досліджень реалізій подібних систем було визначено їх переваги та недоліки і в на основі цього сформовано вимоги до сервера: незалежність від регіону, додавання/видалення/оновлення адресантів, автоматизована відправка листів та автоматизований аналіз повідомлень адресантів.

Програма була написана в інтегрованому середовищі розробки Intellij Idea на мові Java. Було проаналізовано фреймоврки, які підходять для вирішення подібних задах на мові Java. Реалізація була досягнута завдяки фреймворку Spring.

В основу обміну даними з клієнтом було покладено шаблон Модель (Model) – Вигляд (View) – Контролер (Controller), а реалізація внутрішньої архітектури була зроблена на основі шаблону Контролер (Controller) – Сервіс (Service) – Репозиторій (Repository). Цей вибір було зроблено завдяки гнучкості, простоті та перевіреній роками надійність.

Відповідно до необхідного функціоналу та обраної архітектури було розроблено сервер. Протестовано всі частини сервера та перевірено, що всі вони працюють належним чином. При спробі доступу без авторизації та введенні даних в неправильному форматі, сервер продовжує працювати та відповідає на це кодами помилок, що в свою чергу дає змогу без проблем знайти помилку. При введенні правильних даних та у правильному форматі, користувач отримує відповідь та може продовжувати працювати з сервером.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. M. Pollack, O. Gierke, T. Risberg, J. Brisbin та M. Hunger, «Spring Data (First ed.),» New York, O'Reilly, 2013, p. 316.

2. R. Obe, L. Hsu, PostgreSQL: Up and Running, 2012, p. 146/

3. H.Schonig, Mastering PostgreSQL 11: Expert Techniques to Build Scalable, Reliable, and Fault-tolerant Database Applications (2nd Edition), p. 435.

4. Prabath Siriwardena, Mastering Apache Maven 3, p. 498.

5. C. Schaefer, C. Ho, I. Cosmina, R. Harrop, Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools, p. 866.

6. M. Fisher, M. Abbott, The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise, p. 945.

7. C. Scarioni, M. Nardone, Pro Spring Security: Securing Spring Framework 5 and Boot 2-based Java Applications, p .424.

8. P. Mularien, M. Knutson, R. Winch, Spring Security (Third Edition), p. 542.

9. Felipe Gutierrez, Pro Spring Boot 2: An Authoritative Guide to Building Microservices, Web and Enterprise Applications, and Best Practices, p. 520.

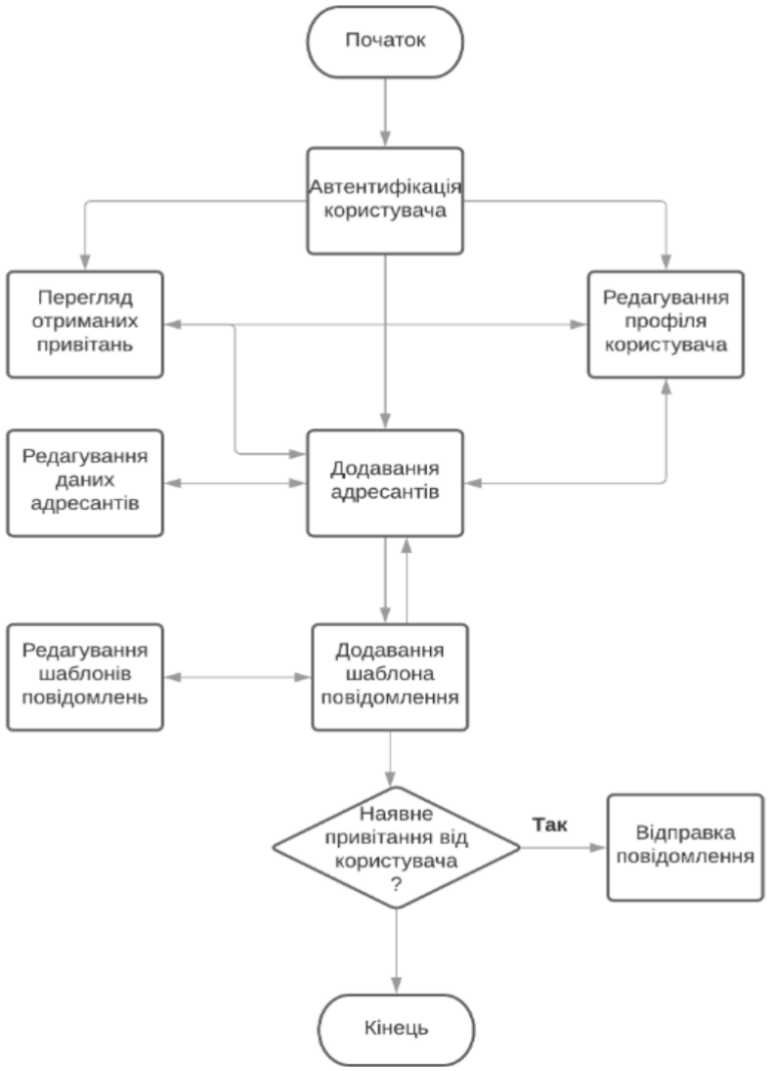
10. Amuthan G, Spring MVC Beginner's Guide: Your Ultimate Guide to Building a Complete Web Application Using All the Capabilities of Spring MVC, p. 554.

ДОДАТОК А

«Система автоматизованого формування й аналізу повідомлень»

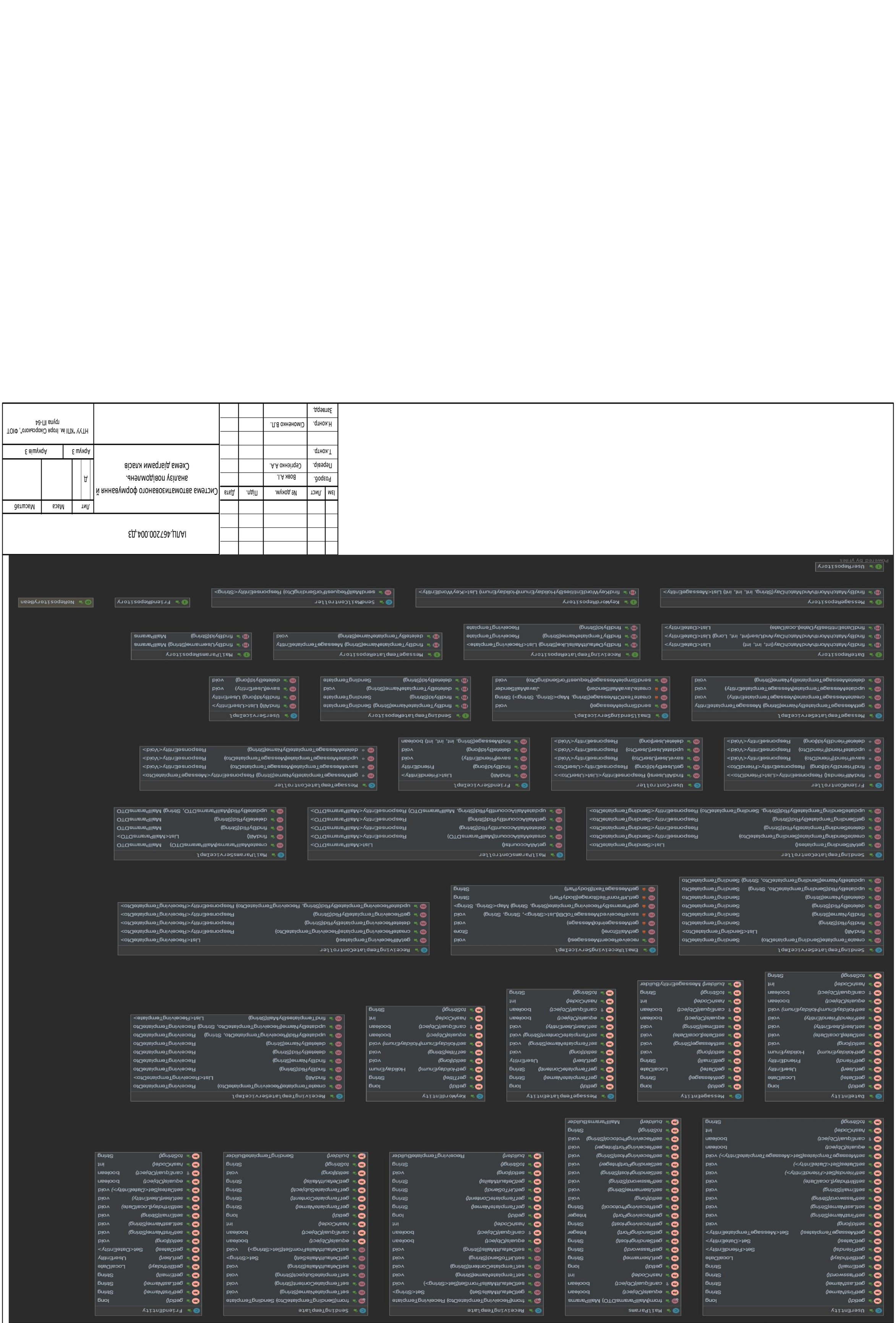
КОПІЇ ГРАФІЧНИХ МАТЕРІАЛІВ

Аркушів 1



					ІАЛЦ.467200.004.Д1	
Ізм.	Лист	№ докум.	Підп.	Дата		
Розроб.		Вовк А.І.			Система автоматизованого формування й аналізу повідомлень. Схема алгоритму користування системою	Лит: Д; Маса: ; Масштаб:
Перевір.		Сергієнко А.А.				Аркуш 1; Аркушів 3
Т.контр.						НТУУ "КПІ ім. Ігоря Сікорського", ФІОТ група ІП-64
Н.контр.		Сімоненко В.П.				
Затверд.						

					ІАЛЦ.467200.004.Д2	
						Лит / Маса / Масштаб
Ізм	Лист	№ докум.	Підп.	Дата	Система автоматизованого формування й аналізу повідомлень. Схема бази даних додатку	Д
Розроб.		Вовк А.І.				
Перевір.		Сергієнко А.А.				Аркуш 2 / Аркушів 3
Т.контр.						
Н.контр.		Сімоненко В.П.			НТУУ "КПІ ім. Ігоря Сікорського", ФІОТ група ІП-64	
Затверд.						



Ізм	Лист	№ докум.	Підп.	Дата
Розроб.		Вовк А.І.		
Перевір.		Сергієнко А.А.		
Т.контр.				
Н.контр.		Симоненко В.Л.		
Затверд.				

ІАЛЦ.467200.004.ДЗ

Система автоматизованого формування й аналізу повідомлень. Схема діаграми класів

Літ	Маса	Масштаб
А		

Аркуш 3 Аркушів 3

НТУУ "КПІ ім. Ігоря Сікорського", ФІОТ
група ІП-64